A PRACTICAL AND FLEXIBLE ENVIRONMENT FOR ADAPTIVE KNOWLEDGE AND DATA FUSION APPLICATIONS.

David Tuck, Imaging & Sensing Team, Industrial Research Ltd, NZ, and
Qun Song, Nik Kasabov & Michael Watts, Dept. Information Science, Univ. Otago, NZ.

ABSTRACT

The purpose of this paper is to describe and demonstrate a new approach for a practical and flexible environment for implementing knowledge and data fusion applications. Data fusion is used today in many engineering and managerial applications to help resolve complex planning, control and optimisation problems. A time-series application case study is presented and discussed: a prototype robot arm control example, utilising a fuzzy neural network (FuNN) tool module for off-line learning and rule manipulation, and a new on-line evolving fuzzy neural network (EFuNN) adapting tool module, from this environment.

1. INTRODUCTION

An Artificial Neural Network (ANN) can provide the ability to produce a model for the mechanisms underlying the information in data sources used in decision-making and time-series prediction processes. However hybrid neuro-fuzzy systems that include both learning from data and fuzzy rules manipulation, add much more to this useful property [1-4]. One particular example for combining neural networks and fuzzy systems is the concept of a Fuzzy Neural Network (FNN) [1, 3]. By fuzzifying an ANN, the quantisation of the inputs and outputs through the application of membership functions, extra robustness is provided when used with redundant, noisy or incomplete input data. Further, this fuzzification technique can provide the means for extracting the information learnt in the form of rules. It is also now possible to add *a priori* knowledge constraints to the network.

Here, two types of FNNs are demonstrated as part of a hybrid software environment: the Fuzzy Neural Network (FuNN) [1, 4-7], used for off-line learning rule manipulation, and the Evolving Fuzzy Neural Network (EFuNN) [8-10] used for on-line real time learning and prediction. The latest environment FuzzyCOPE/3, is a suite of data processing and neural network tools for Microsoft Windows, see http://kel.otago.ac.nz/software/FuzzyCOPE3/.

2. THE FUZZY NEURAL NETWORKS

FNNs are connectionist models for fuzzy rules implementation and inference [1, 3, 5-7]. However, there are a wide variety of architectures and functionality, differing in the type of fuzzy rules, type of inference method, and modes of operation. In general the architecture of these FNNs consist of five layers, Figure 1. These layers in order, input to output, are:

A. An input layer, where the neurones represent the linguistic variables of the input data;
B. A fuzzification, or conditioning layer, where the neurones represent the fuzzy values;
C. A rules layer, where the neurone nodes represent the fuzzy rules;
D. An action layer, where the neurones represent the fuzzy values of the output variables;
E. A final output layer, where the neurones represent the output linguistic variables.

The particular example illustrated below in Fig.1 has two inputs, with two fuzzy membership functions (MF) each, two rule nodes, and two outputs, again with two MF each.

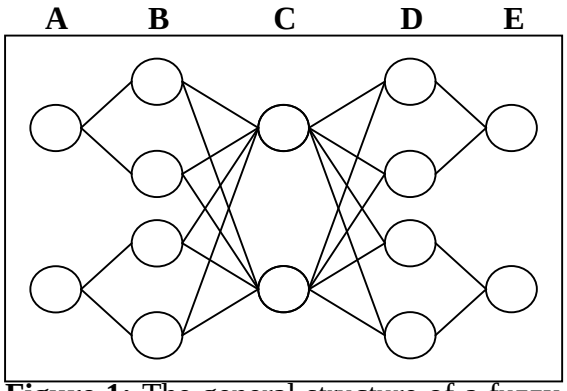

Figure 1: The general structure of a fuzzy neural network.

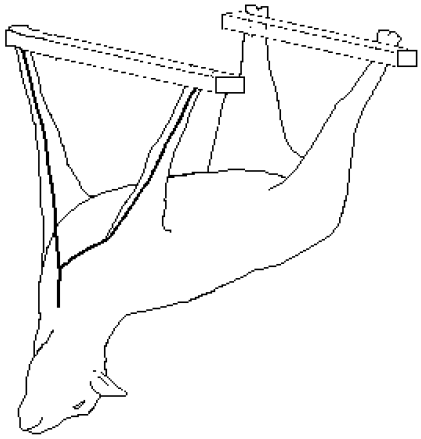

Figure 2: An example of the sheep carcass Y-cut path.

The FuNN module used here is characterised by the following features. It uses weighted fuzzy rules [1], modified back-propagation algorithms for training that include training with forgetting, genetic algorithms to improve training, training with or without modifying the membership functions [7], different types of rule extraction [6, 7], and rule insertion. Usually, the FuNN employs standard triangular membership functions and the number of rule nodes and rules are defined and fixed by the analyst prior to initialisation. However, FuNNs do have some difficulties when applied to on-line processing [14], but these can be overcome by the recently developed evolving EFuNN [8-10].

The EFuNN is a new on-line learning tool where the input sources of information are not pre-defined and can vary during the on-line learning process, thus allowing for "on the fly" fusion of different sources of information and fuzzy rules. It is available from http://kel.otago.ac.nz/software. In this EFuNN architecture, the network begins with an empty rule layer. As training patterns are presented to the network, examples that are not adequately represented by the rule layer, trigger the addition of nodes to represent them and an update of the present connection weights. Each rule node, after training, therefore represents one or several training examples. EFuNNs exhibit:

- Memory-based learning where exemplars of data are stored as they arrive at the inputs;
- Open structure – the number of the inputs and the outputs of the EFuNN can be varied, making fusion of an unknown number of sources possible on-line and "on the fly";
- Rapid, one pass training with good generalisation capability, both local and global and;
- Robustness to forgetting, local tuning of weights, and rapid adaptation to new data.

3. CASE STUDY – ON-LINE ROBOT CONTROL

3.1 The Problem and Solution Approach

In New Zealand meat-works, lambs are valued for both their export pelt and meat products. In order to remove the carcass pelt without damage to itself or the flesh underneath and hygiene, extreme care is required in the initial cutting operation of the skin. For this example, a new robot cutting path planner approach has been investigated. In the past an algorithmic path planning robotics system was developed and has shown great potential over the traditional manual butchering preparation. However, this approach is limited by the rather restricted semi-automated algorithmic implementation developed. We started to

explore the use of the FuNN tool from FuzzyCOPE/3 to first develop a model [4] of this current algorithmic planner. Now, with that model demonstrating very encouraging results (Part One, in 3.3), we have also begun pursuing the use of the on-line adaptation properties of the EFuNN (Part Two, in 3.4). This will allow the model to continue adapting to the highly variable sizes and shapes of this natural product (sheep).

3.2 Experimental Data Sets

The carcass de-pelting process is performed on the sheep carcass while hanging upside down on a moving conveyor chain, Figure 2. This skin only Y-cut, is really two separate cuts, and begins with one front leg at the hoof following down that leg and across the lower neck/chest region of the carcass (also known as the brisket) and terminates just past the midline of the body. A second cut is then carried out following a similar path from the second front hoof, but mirroring the path of the first cut to finish just past the point of intersection with the first cut. When the Y- cut is performed correctly, the pelt can be pulled off the carcass as a whole piece, with minimal damage and contamination.

For this time-series study, the sheep carcass Y-cut sensor data was used, together with the algorithm path data, for training with a fixed parameter setting. Three sensors provide 3-D measurements between important points on the carcass so that the robotic skin cutting operation can be planned. These measurements are; the separation between the two front hooves, the highest point on the brisket, and finally the horizontal offset between the brisket and the trachea region of the neck. With the carcass hung by its hooves, the x_0, y_0, z_0 starting points are easily identified and provide a fixed [0, 0, 0] 3-D reference for the start of the cut. However, the meeting point of the cuts and their paths down the front legs of the carcass are very dependent on the size and breed of the animal, determined by the sensor data.

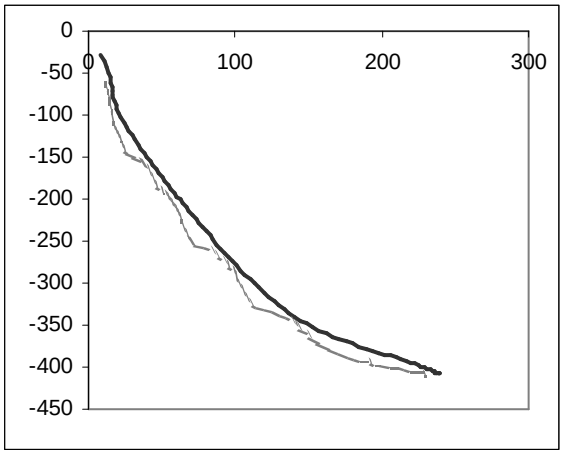

Figure 3: Typical x-y cutting path (mm) from FuNN.

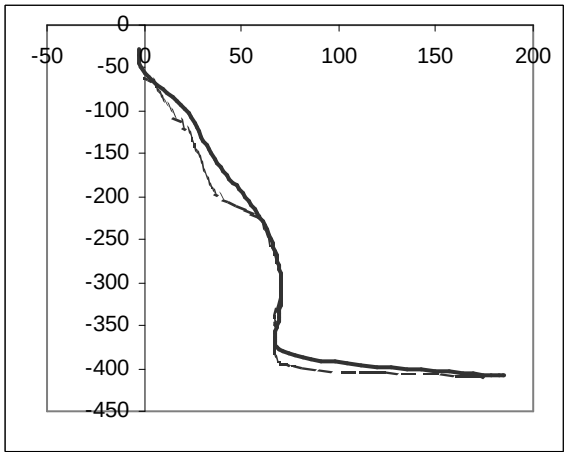

Figure 4: Resultant FuNN z-y cutting path (mm) for Figure 3.

3.3 Training Path Planning with FuNN – Part One Results

An off-line cutting path planning model has been developed using FuNN to predict the next knife position for time, t. The best results obtained have been with an 18 node rule layer and after only 100 training epochs. Figures 3 and 4 display a typical prediction result of a single cut on one sheep, x versus y and z versus y respectively, with the actual (black line) and the FuNN predicted (grey line) cutting paths superimposed. The average RMS differences are

3.3, 6.1 and 4.4 (mm) for the x(t), y(t), and z(t) for the target-to-predictions values after training from 49 carcass data, and 4.0, 10.8 and 10.9 (mm) for the 33 test data respectively.

3.4 Training Path Planning with EFuNN – Part Two Results

On-line prediction models were next developed using an EFuNN. This was first created with the same number of inputs (18) and three outputs (a total of 90 input and 27 output MFs), as the FuNN result in 3.3, with only a single rule node, because EFuNNs add rule nodes as required. The network was trained off-line [4] for one sheep data case and then was ready to predict and learn on-line the correct new x, y, z values. By using the actual target output values, each different input–output association was added to the EFuNN after the last prediction point adaptation step. When the maximum number of rule nodes specified was reached, the EFuNN began pruning nodes (see [4, 8-10]) before continuing. This cycle was repeated for the rest of the 81 sheep examples, to predict the cutting paths on-line.

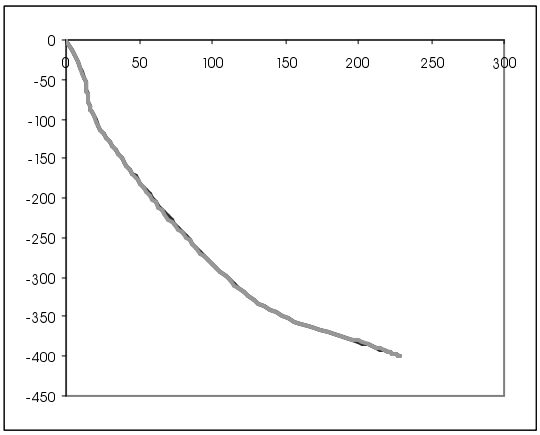

Figure 5: Typical x-y cutting path (mm) from EFuNN.

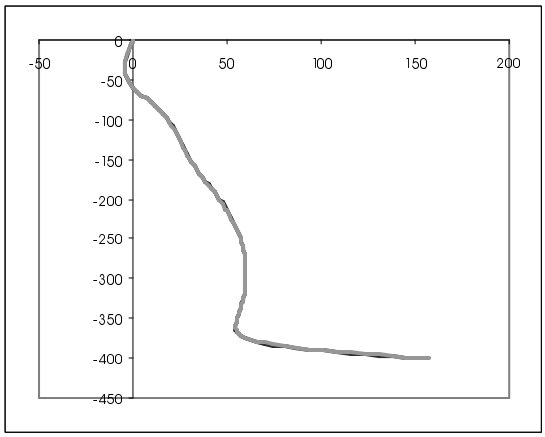

Figure 6: Resultant EFuNN z-y cutting path (mm) for Figure 5.

Figures 5 and 6 display a typical prediction result for the EFuNN for a single cut on one sheep, x versus y and z versus y respectively, with the actual (black line) and the predicted (grey line) cutting paths superimposed. Trials starting with different sheep produced EFuNNs with an average of 41 rule nodes. The overall average RMS differences were 0.08, 0.45 and 0.10 (mm) for the target-to-predicted x(t), y(t), and z(t) values of the remaining 81 test data.

3.5 Comparative Analysis of the Different Fusion Techniques

Both the FuNN and EFuNN were able to approximate the data to a reasonable degree of accuracy. However, while the FuNN required 100 training epochs of 49 sheep, the EFuNN required only a single pass through one sheep's data, to produce a result that was far better for the remaining 81 sheep. It is this rapid prediction and adaptation capability that is one of the major advantages of EFuNNs. Rules from EFuNN can also be extracted and inserted [8-10].

4. DISCUSSION AND CONCLUSIONS

The results here demonstrate the potential of FuzzyCOPE/3 as a fusion environment for providing solutions to previously difficult to-solve-problems. To date it is used by more than

35 universities for teaching, and more than 200 developers of intelligent information systems, worldwide. We hope this paper promotes awareness of this practical and versatile data fusion environment, and entices others to investigate and apply it to new real world problems.

Fuzzified connectionist-based algorithms are robust when the appropriate techniques are used. They allow the analyst to learn relationships between the input and output variables without making assumptions about the data distribution. This can improve the prediction or classification accuracy based on updating the transfer function and not manipulating the incoming data flow. Also FNNs require fewer training examples, especially "evolving" FNNs, than traditional ANN sensor data fusion methods. FNNs can be shown to have a distinct advantage [1], over fuzzy rules and more traditional statistical methods, especially here where the prediction outputs follow a very non-linear path such as demonstrated above. Finally, adaptive learning algorithms enable the EFuNNs to learn relationships between input data and output data in an iterative way [8-10] and on-line, and fuzzy rules may then be extracted to explain what the network has learned.

In this paper, two of the hybrid neuro-fuzzy modules (FuNN and EFuNN) were successfully applied to a multi-sensor fusion problem. Further development of this robotic path planning problem, using the better EFuNN tool, will continue to developed a fully automated solution. The next stage in this work will develop a second EFuNN model that can generate the first five cutting path coordinates for the (t-5) to (t-1) inputs to start off our latest model.

Acknowledgments

This research is partially supported by the Foundation for Research, Science and Technology, New Zealand through a grant UOO808 to the University of Otago, and NSOF at Industrial Research Limited. Also thanks must go to Dr Malcolm Taylor for the acquisition and generation of the robotic cutting path data used in the case study.

References

1. Kasabov, N. *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering.* The MIT Press, CA, MA, 1996.
2. Tuck, D. Forecasting a Variable Multiple Cyclic Process with Confidence. *Proc. ICONIP'97, ANZIIS'97 & ANNES'97*, New Zealand, Springer, 1997, Vol.2, pp992-995.
3. Yamakawa, T; Kusanagi, H;. Uchino, E and Miki, T. A New Effective Algorithm for Neo Fuzzy Neuron Model. In: *Proc. Fifth IFSA World Congress*, 1993, pp1017-1020.
4. Kasabov, N; Tuck, DL & Watts, M. Implementing Knowledge and Data Fusion in a Versatile Software Environment for Adaptive Learning and Decision-Making. *Proc. 2nd Int. Conf. on Information Fusion (FUSION'99*), CA, Omnipress, 1999, Vol.1, pp455-462.
5. Kasabov, N. Adaptable Connectionist Production Systems. *Neurocomputing*, 13 (2-4), 1996, pp95-117.
6. Kasabov, N. Learning Fuzzy Rules and Approximate Reasoning in Fuzzy Neural Networks and Hybrid Systems. *Fuzzy Sets and Systems*, 82 (2), 1996, pp2-20.
7. Kasabov, N; Kim, J; Watts, M. and Gray, A. Architecture for Adaptive Learning and Knowledge Acquisition. *Information Sciences*, 101 (3-4), 1997, pp155-175.
8. Kasabov, N. Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation. In: Yamakawa and Matsumoto (eds), *Methodologies for the Conception, Design and Application of Soft Computing*, World Scientific, 1998, pp271-274.
9. Kasabov, N. ECOS: A Framework for Evolving Connectionist Systems and the ECO Learning Paradigm. *Proc. of ICONIP'98*, Japan, IOS Press, 1998, pp1222-1235.
10. Kasabov, N. The ECOS Framework and the ECO Learning Method for Evolving Connectionist Systems. *J. Advanced Computational Intelligence*, 2 (6), 1998, pp1-8.