The Utility Driven Dynamic Error Propagation Network

A J Robinson F Fallside

CUED/F-INFENG/TR.1 (1987)

Part of this document has been accepted for presentation at the IEEE Conference on "Neural Information Processing Systems - Natural and Synthetic", Denver, 1987.

The Utility Driven Dynamic Error Propagation Network

A J Robinson, F Fallside
CUED/F-INFENG/TR.1 (1987)
Cambridge University Engineering Department, England

November 4, 1987

Abstract

Error propagation networks are able to learn a variety of tasks in which a static input pattern is mapped onto a static output pattern. This paper presents a generalisation of these nets to deal with time varying, or dynamic patterns. Three possible architectures are explored which deal with learning sequences of known finite length and sequences of unknown and possibly infinite length. Several examples are given and an application to speech coding is discussed.

A further development of dynamic nets is made which allows them to be trained by a signal which expresses the correctness of the output of the net, the utility signal. One possible architecture for such a utility driven dynamic net is given and a simple example is presented. Utility driven dynamic nets are potentially able to calculate and maximise any function of the input and output data streams, within the considered context. This is a very powerful property, and an appendix presents a comparison of the information processing in utility driven dynamic nets and that in the human brain.

Contents

1 Introduction 1

2 Static Error Propagation Nets 1

3 Dynamic Error Propagation Nets 3

3.1 Development from Linear Control Theory 3

3.2 Architectures 4

3.2.1 The Finite Impulse Response (FIR) Dynamic Net 4

3.2.2 The Infinite Impulse Response (IIR) Dynamic Net 6

3.2.3 The State Compression Dynamic Net 7

3.3 Some Examples of Dynamic Nets 8

3.3.1 Time Delay 8

3.3.2 Bistable 9

3.3.3 Movement Detection without Wraparound 9

3.3.4 Movement Detection with Wraparound 10

3.3.5 Letter to Word Conversion 11

3.4 Application to Speech Coding 12

3.4.1 The Architecture of a General Coder 12

3.4.2 Training of the Speech Coder 13

3.4.3 Quantitative Comparison of Performance 13

3.4.4 Visual Comparison of Performance 14

3.4.5 Audible Comparison of Performance 14

3.5 Limitations of Dynamic Nets 16

4 Utility Driven Dynamic Nets 17

4.1 Architectures 17

4.2 An Example Utility Driven Dynamic Net 18

5 Conclusion 19

References 20

A Appendix: Real and Artificial Intelligence 21

A.1 Levels of Isomorphism 21

A.2 Hardware Organisation 22

A.3 Learning 22

A.4 Memory 23

A.5 Recognition 24

A.6 Thought 24

A.7 Consciousness 25

A.8 Holistic Processing 25

1 Introduction

This report develops a powerful new form of connectionist net, the utility driven dynamic net. Connectionist nets were popular in the 1960's, initiated by the perceptron (Rosenblatt, 1958, 1962) which showed great promise as a result of an efficient learning algorithm known as the 'perceptron convergence procedure'. In 1969 it was shown that the perceptron was unable to learn certain tasks such as parity and connectedness (Minsky and Papert, 1969) and this resulted in a decline in their popularity. Recently there has been a resurgence of interest through the discovery of new learning algorithms and the continued increase in available processing power. In particular the error propagation algorithm (Rumelhart, Hinton and Williams, 1986) has extended the perceptron to more than one layer, so overcoming the previous learning difficulties. A review of current connectionist interest is given in 'Parallel Distributed Processing' (Rumelhart and McClelland, 1986).

The error propagation algorithm is taken as a starting point. This algorithm can be used to train static nets which can make arbitrary mappings from input to output, but have no memory for past inputs. An extension to the static net is developed, the dynamic net, which feeds back part of the output to the input, so creating some internal storage and allowing a far greater class of problems to be learned. To illustrate the power of this extension several problems are considered, starting with basic logic and culminating in an application to speech coding.

These dynamic nets are then developed further. Instead of learning by presentation of the input and the desired output, the net computes a likely output and is given a training signal, the utility signal, which indicates if the output is correct. Thus the machine must first learn the relationship between the observed sequences of inputs and outputs and the utility signal, and then learn to maximise the training signal. A simple example is given.

2 Static Error Propagation Nets

A static net is defined by a set of units and links between the units. Denoting o_i as the value of the i^{th} unit, and $w_{i,j}$ as the weight of the link between o_i and o_j, we may divide up the units into input units, hidden units and output units. If we assign o_0 to a constant to form a bias, the input units run from o_1 up to $o_{n_{\text{inp}}}$, followed by the hidden units to $o_{n_{\text{hid}}}$ and then the output units to $o_{n_{\text{out}}}$. The values of the input units are defined by the problem and the values of the remaining units are defined by:

$$\text{net}_i = \sum_{j=0}^{i-1} w_{i,j} o_j \tag{1}$$

$$o_i = f(\text{net}_i) \tag{2}$$

where $f(x)$ is any continuous monotonic non-linear function and is known as the activation function. The function used in all the examples is:

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{3}$$

These equations define a net which has the maximum number of interconnections. This arrangement is commonly restricted to a layered structure in which units are only connected

to the immediately preceding layer. The examples in this report do not use this popular form, but use a maximally interconnected net without connections between output units. This decision was based on a minimal constraint design philosophy, if a layered structure is applicable to a problem then the net can disregard the interlayer links (Robinson, 1986). The architecture of these nets is specified by the number of input, output and hidden units. In this report a static net is pictured as a transformation of an input u, to output y, as in figure 1.

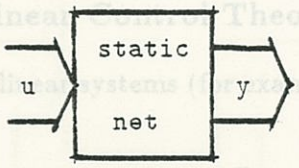


figure 1

The net is trained by using a gradient descent algorithm which minismises an energy term, E, defined as the summed squared error between the actual outputs, o_i, and the target outputs, t_i.

$$E = \frac{1}{2} \sum_{i=n_{\text{hid}}+1}^{n_{\text{out}}} (t_i - o_i)^2 \tag{4}$$

A gradient descent algorithm is used to minimize E. This defines an error signal, δ_i, for each unit:

$$\delta_i = f'(\text{net}_i)(t_i - o_i) \qquad n_{\text{hid}} < i \leq n_{\text{out}} \tag{5}$$

$$= f'(\text{net}_i) \sum_{j=i+1}^{n_o} \delta_i w_{j,i} \qquad n_{\text{inp}} < i \leq n_{\text{hid}} \tag{6}$$

where $f'(x)$ is the derivative of $f(x)$ and t_i is the target output for the i^{th} unit. The error signal is combined with the activations of the units to get the change in each weight, $\Delta w_{i,j}$.

$$\Delta w_{i,j} = \eta \delta_i o_j \tag{7}$$

where η is a constant of proportionality which determines the learning rate. If a small finite example set is used, it is possible to update the weights after the complete set has been presented. In all the examples in this report the training set is either large, or non-recurring, so the weights are updated after every example.

The above equations define the error signal, δ_i, for the input units as well as for the hidden units. Thus any number of static nets can be connected together, the values of δ_i being passed from input units of one net to output units of the preceding net. It is this ability of error propagation nets to be 'glued' together in this way that enables the construction of dynamic nets.

3 Dynamic Error Propagation Nets

The essential quality of the dynamic net is is that its behaviour is determined both by the external input to the net, and also by its own internal state. This state is represented by the activation of a group of units. These units form part of the output of a static net and also part of the input to another copy of the same static net in the next time period. Thus the state units link multiple copies of static nets over time to form a dynamic net.

3.1 Development from Linear Control Theory

The analogy of a dynamic net in linear systems (for example Jacobs, 1974) may be stated as:

$$x_{t+1} = Ax_t + Bu_t \quad (8)$$

$$y_t = Cx_t \quad (9)$$

where u_t is the input vector, x_t the state vector, and y_t the output vector at the integer time t. A, B and C are matrices.

The structure of the linear systems solution may be implemented as a non-linear dynamic net by substituting the matrices A, B and C by static nets, represented by the non-linear functions $A[.]$, $B[.]$ and $C[.]$. The summation operation of Ax_t and Bu_t could be achieved using a net with one node for each element in x and u and with unity weights from the two inputs to the identity activation function $f(x) = x$. Alternatively this net can be incorporated into the $A[.]$ net giving the architecture of figure 2.

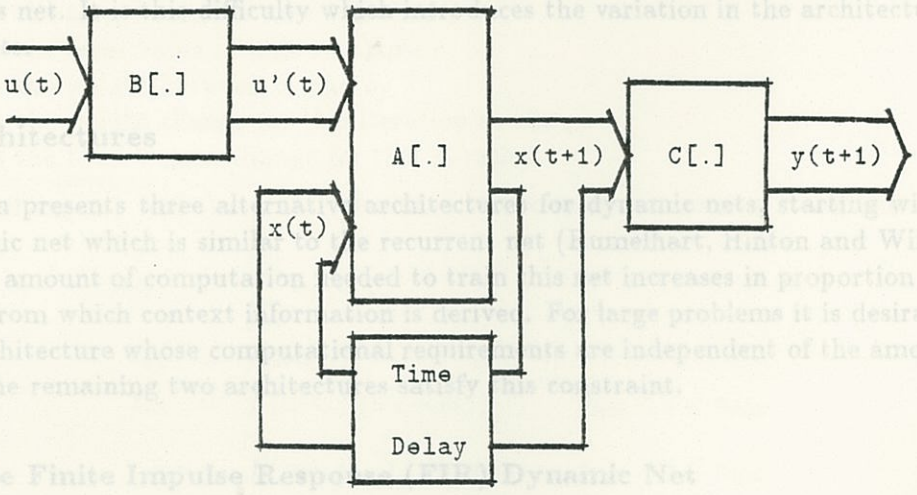


figure 2

The input is coded by net $B[.]$ and then the output is fed into $A[.]$ along with the previous output of $A[.]$ and the resulting output is passed through $C[.]$ to yield the overall output of the system. The three networks of the previous dynamic net architecture may be combined into one, as in figure 3. Simplicity of architecture is not just an aesthetic consideration. If

three nets are used then each one must have enough computational power for its part of the task, combining the nets means that only the combined power must be sufficient and it allows common computations can be shared.

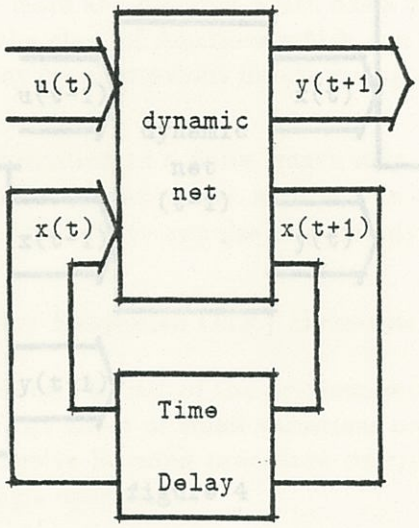


figure 3

The error signal for the output y_{t+1}, can be calculated by comparison with the desired output. However, the error signal for the state units, x_t, is only given by the net at time $t+1$, which is not known at time t. Thus it is impossible to use a single backward pass to train this net. It is this difficulty which introduces the variation in the architectures of dynamic nets.

3.2 Architectures

This section presents three alternative architectures for dynamic nets, starting with the FIR dynamic net which is similar to the recurrent net (Rumelhart, Hinton and Williams, 1986). The amount of computation needed to train this net increases in proportion to the time span from which context information is derived. For large problems it is desirable to have an architecture whose computational requirements are independent of the amount of context. The remaining two architectures satisfy this constraint.

3.2.1 The Finite Impulse Response (FIR) Dynamic Net

If the output of a dynamic net, y_t, is dependent on a finite number of previous inputs, $u_{t-\tau}$ to u_t, or if this assumption is a good approximation, then it is possible to formulate the learning algorithm by expansion of the dynamic net for a finite time, as in figure 4.

Consider only the component of the error signal in past instantiations of the nets which is the result of the error signal at time t. The error signal for y_t is calculated from the target output and the error signal for x_t is zero. This combined error signal is propagated back

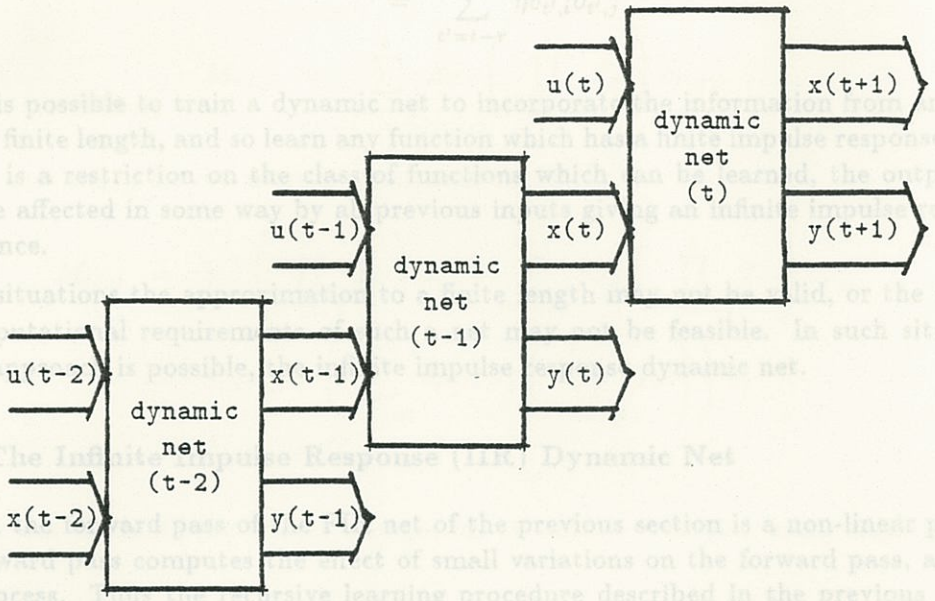


figure 4

though the dynamic net at t to yield the error signals for u_t and x_t. Similarly these error signals can then be propagated back through the net at $t-1$, and so on for all relevant inputs. The summed error signal is then used to change the weights as for a static net.

Formalising the FIR dynamic net:

$o_{t,i}$	is the output value of unit i at time t
$t_{t,i}$	is the target value of unit i at time t
$\delta_{t,i}$	is the error value of unit i at time t
$w_{i,j}$	is the weight between o_i and o_j
$\Delta w_{t,i,j}$	is the weight change for this iteration at time t
$\Delta w_{i,j}$	is the total weight change for this iteration

The values of $o_{t,i}$, $\delta_{t,j}$ and $\Delta w_{i,j}$ are calculated in the same way as in a static net.

$$\text{net}_{t,i} = \sum_{j=0}^{i-1} w_{i,j} o_{t,j} \tag{10}$$

$$o_{t,i} = f(\text{net}_{t,i}) \tag{11}$$

$$\delta_{t,i} = f'(\text{net}_{t,i})(t_{t,i} - o_{t,i}) \qquad n_{\text{hid}} < i < n_{\text{out}} \tag{12}$$

$$= f'(\text{net}_{t,i}) \sum_{j=i+1}^{n_o} \delta_{t,j} w_{j,i} \qquad n_{\text{inp}} < i \leq n_{\text{hid}} \tag{13}$$

$$\Delta w_{t,i,j} = \eta \delta_{t,i} o_{t,j} \tag{14}$$

The total weight change is given by the summation of the partial weight changes for all previous times.

$$\Delta w_{i,j} = \sum_{t'=t-\tau}^{t} \Delta w_{t',i,j} \tag{15}$$

$$= \sum_{t'=t-\tau}^{t} \eta \delta_{t',i} o_{t',j} \tag{16}$$

Thus, it is possible to train a dynamic net to incorporate the information from any time period of finite length, and so learn any function which has a finite impulse response. Note that this is a restriction on the class of functions which can be learned, the output will always be affected in some way by all previous inputs giving an infinite impulse response performance.

In some situations the approximation to a finite length may not be valid, or the storage and computational requirements of such a net may not be feasible. In such situations another approach is possible, the infinite impulse response dynamic net.

3.2.2 The Infinite Impulse Response (IIR) Dynamic Net

Although the forward pass of the FIR net of the previous section is a non-linear process, the backward pass computes the effect of small variations on the forward pass, and is a linear process. Thus the recursive learning procedure described in the previous section may be compressed into a single operation.

Given the target values for the output of the net at time t, equation 12 defines values of $\delta_{t,i}$ at the outputs. If we denote this set of $\delta_{t,i}$ by D_t then equation 13 states that any $\delta_{t,i}$ in the net at time t is simply a linear transformation of D_t. Writing the transformation matrix as S:

$$\delta_{t,i} = S_{t,i} D_t \tag{17}$$

In particular the set of $\delta_{t,i}$ which is to be fed back into the network at time $t-1$ is also a linear transformation of D_t

$$D_{t-1} = T_t D_t \tag{18}$$

or for an arbitrary time t':

$$D_{t'} = \left(\prod_{t''=t'+1}^{t} T_{t''} \right) D_t \tag{19}$$

so substituting this into equation 16:

$$\Delta w_{i,j} = \eta \sum_{t'=-\infty}^{t} S_{t',i} D_{t'} o_{t',j} \tag{20}$$

$$= \eta \sum_{t'=-\infty}^{t} S_{t',i} \left(\prod_{t''=t'+1}^{t} T_{t''} \right) D_t o_{t',j} \tag{21}$$

which can be rewritten as:

$$\Delta w_{i,j} = \eta M_{t,i,j} D_t \tag{22}$$

where:

$$M_{t,i,j} = \sum_{t'=-\infty}^{t} S_{t',i} \left(\prod_{t''=t'+1}^{t} T_{t''} \right) o_{t',j} \tag{23}$$

and note that $M_{t,i,j}$ can be written in terms of $M_{t-1,i,j}$:

$$M_{t,i,j} = S_{t,i} \left(\prod_{t''=t+1}^{t} T_{t''} \right) o_{t,j} + \sum_{t'=-\infty}^{t-1} S_{t',i} \left(\prod_{t''=t'+1}^{t} T_{t''} \right) o_{t',j} \tag{24}$$

$$= S_{t,i} o_{t,j} + \left(\sum_{t'=-\infty}^{t-1} o_{t',j} S_{t',i} \left(\prod_{t''=t'+1}^{t-1} T_{t''} \right) \right) T_t \tag{25}$$

$$= S_{t,i} o_{t,j} + M_{t-1,i,j} T_t \tag{26}$$

Hence we can calculate the weight changes for an infinite recursion using only the finite matrix M.

If this approach is to be of practical use, we must consider the overall storage and computation requirements as compared with the FIR net. The net has n_{out} units of which $n_{\text{tar}} = n_{\text{out}} - n_{\text{hid}}$ are output units, the IIR net requires $n_{\text{out}} n_{\text{tar}}$ locations for storage of the M matrix. If a context of t time slots is required to solve the problem then the FIR net requires $t n_{\text{out}}$ locations. Thus the FIR net requires less storage for $t < n_{\text{tar}}$.

The computational requirements for the FIR net are given in equations 15 and 16. Designating i_{back} as the number of instructions needed to compute a single backward pass of the net, as in equation 15, then $t i_{\text{back}}$ instructions must be executed to compute the weight changes per iteration.

The weight changes for the IIR net are given by equations 22 and 26. $M_{t,i,j}$ and D_t are matrices of order n_{tar} and so equation 22 requires order n_{tar} multiplications per weight, a total of $n_{\text{tar}} i_{\text{back}}$ computations. Equation 26 first requires order of $n_{\text{tar}} i_{\text{back}}$ computations to calculate S_t, and then order of n_{tar}^2 computations to compute $M_{t-1,i,j} T_t$ and order n_{tar} computations for $o_{t,j} S_{t,i}$, each of which must be computed i_{back} times. Taking the highest order, the IIR net requires order $n_{\text{tar}}^2 i_{\text{back}}$ computations.

In comparison, the IIR net is computationally more efficient if $t > n_{\text{tar}}^2$. As the examples in this report use values of t in the range 1 to 4, and n_{tar} in the range 1 to 17, the IIR net is not computationally efficient and has not been used in any of the examples.

3.2.3 The State Compression Dynamic Net

The previous architectures for dynamic nets rely on the propagation of the error signal back in time to define the format of the information in the state units. An alternative approach is to use another error propagation net to define the format of the state units. The overall architecture is given in figure 5.

The encoder net is trained to code the current input and current state onto the next state, while the decoder net is trained to do the reverse operation. The translator net codes the next state onto the desired output. This encoding/decoding attempts to represent the current input and the current state in the next state, and by the recursion, it will try to

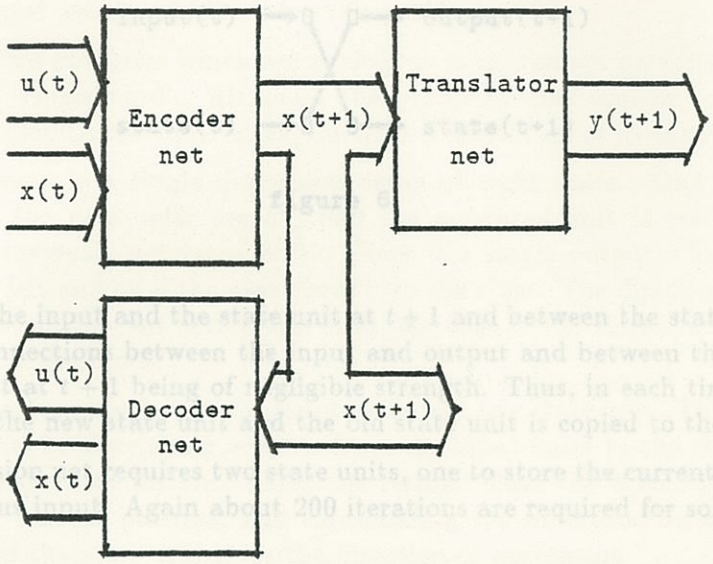


figure 5

represent all previous inputs. Feeding errors back from the translator directs this coding of past inputs to those which are useful in forming the output.

A feature of this architecture is that recent information tends to be stored in the state units whether it is required to compute the output or not. As with the IIR architecture, the amount of computation per learning cycle is not dependent on the time span of the input data.

3.3 Some Examples of Dynamic Nets

This section presents some simple applications of dynamic nets to illustrate their computational power. Two forms of dynamic nets are used, a finite impulse response dynamic net looking back over two time slots and a state compression dynamic net. The activation function, equation 3 has a maximum value of $+1.0$ and a minimum value of -1.0. However, these values can not be achieved with finite input, so the target values of $+0.8$ for high and -0.8 for low were used instead. Except when stated otherwise, the learning rate η was set to 0.1, and the nets were considered to have learned when the short term residual energy fell below 0.01.

3.3.1 Time Delay

One of the simplest problems for a dynamic net is to reproduce a random one bit input after a unit time delay.

The FIR net requires one state unit and no hidden units to learn this task. Figure 6 shows the nodes of this net as circles and the significant weights as solid lines.

The net learns in about 200 iterations by forming connections of approximately unity

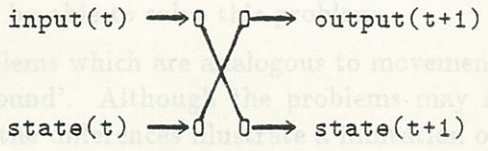


figure 6

strength between the input and the state unit at $t+1$ and between the state unit at t and the output, the connections between the input and output and between the state unit at t and the state unit at $t+1$ being of negligible strength. Thus, in each time slot the the input is copied to the new state unit and the old state unit is copied to the output.
The state compression net requires two state units, one to store the current input and one to store the previous input. Again about 200 iterations are required for solution.

3.3.2 Bistable

Another basic problem for dynamic nets is to oscillate between states with no external input. An FIR net may learn to do this using no input units, one state unit and one output unit, as in figure 7.

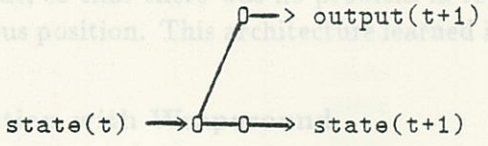


figure 7

With the signed activation function, a negative weight between the state input and output reverses the sign of the state and the sign of the weight to the output unit ensures the desired phase. This net learns in about 250 iterations.

The state compression dynamic net is not able to learn this problem as it stands. With no information in the input units, the energy of the encoder/decoder part of the net is minimized by setting the state units to zero. Zeroed state units also contain no information and so the desired output can not be derived from these units. This may be overcome by presenting an input which is uncorrelated with the output. If a one bit binary input is fed to a net with eight state units, the bistable problem may be learned in about 500 iterations.

3.3.3 Movement Detection without Wraparound

The movement detection problem was inspired by neurobiological research that showed that the human brain contains single cells which can detect movement (Poggio and Koch,

1987). If dynamic error propagation nets are sufficiently powerful to model the human brain, then a net must also be able to solve this problem.

This is the first of two problems which are analogous to movement detection on a 'retina' with and without 'wraparound'. Although the problems may appear to be of similar computational complexity, the differences illustrate a limitation of dynamic nets.

The input to the nets is a single dimension array of eight units. One of these units is activated whilst the remainder are off, and the activated unit is restricted to be a neighbour of the previously activated unit. There is a single output which is on if the movement is to the left and off if the movement is to the right. The direction of movement is random, except when the last active unit is at one edge of the retina in which case the movement is towards the centre.

The FIR net architecture involved eight input units and one state unit. This net learns in about 5300 iterations by developing weights from the input units to the state unit which monotonically increase with distance along the retina. Thus the activation of the state unit represents the current position, and thresholding the difference between this value and the last value of the state unit gives the direction of movement.

Two architectures of state compression nets were considered, one with two and one with nine state units. Two state units represents the minimum required to solve the problem, one to store the present position and one to store the previous position. The first version failed to learn in 2^{16} iterations, presumably because the input contains noise from the random directions and there is no spare information capacity in the state units to record this and still be able to solve the problem. The second architecture was designed to have one state unit for every input, so that there was no problem in replicating the input, and one more to store the previous position. This architecture learned in about 1000 iterations.

3.3.4 Movement Detection with Wraparound

The problem of movement detection with wraparound is the same as the case without wraparound, except that an attempt to move off on end of the retina results in the active unit appearing at the other end. This generates some problems as the comparison of two scalars can no longer be used to judge the direction of motion.

The FIR net with eight inputs, sixteen hidden units, sixteen state units and one output unit learns all but two transitions in about 6000 iterations. However the final transition requires a considerable movement in weight space for a small decrease in energy. In learning the remainder of the problem some activations became very large, reducing the slope of these units to near zero and so blocking the error signal from propagating back through them. The final transition was not learned.

The state compression net had the same number of input, state and hidden units as the FIR net. This net did not suffer from the same instabilities and learned in 1500 iterations. Thus this state compression net gives a means of solving a problem which can not be solved by the FIR net.

3.3.5 Letter to Word Conversion

As an example of sequence recognition, the constituent letters of words were presented sequentially to a dynamic net and an output unit corresponding to the word was activated upon its completion. A connectionist solution to this problem has already been formulated with predefined weights (Tank and Hopfield, 1987) and this example shows that it is also possible to learn the weights of a connectionist network which solves this problem.

The nets had 26 inputs (one for each letter of the alphabet), 34 hidden units, 34 state units and 8 output units, one for each of the unique words in "the quick brown fox jumped over the lazy dog". One word was chosen at random and the letters presented sequentially to the net by activating one input unit and switching the rest off. The desired output was for all units to be off until the input after the completion of the word, when one output is activated. The letters of succeeding words were run together without punctuation, so the net had to learn to segment the letters into words and then label the segments. Example input is given in table 1.

time	input letter activated	output word activated
t-3	.	.
t-2	.	.
t-1	.	.
t	z	none
t+1	y	none
t+2	q	lazy
t+3	u	none
t+4	i	none
t+5	c	none
t+6	k	none
t+7	d	quick
t+8	o	none
t+9	.	.
t+10	.	.
t+11	.	.

table 1

The time between repetitions of input-output pairs is considerably longer in this example than any of the previous examples, and the learning rate was correspondingly reduced to $\eta = 0.05$ to compensate.

Several versions of the FIR architecture were used, differing in the number of previous letters of context that were considered. When the context was limited to the last letter, or to the last two letters, this net could not resolve the ambiguities, and the net did not learn. With three letter context the FIR net learned in 17,000 iterations and with four letter context only 7,500 iterations were required to solution.

The state compression net required the presentation of 15,000 iterations before reducing its energy to below the threshold of 0.01.

3.4 Application to Speech Coding

The problem of speech coding is one of finding a suitable model to remove redundancy and hence reduce the data rate of the speech. The Boltzmann machine learning algorithm has already been extended to deal to the dynamic case and applied to speech recognition (Prager, Harrison and Fallside, 1986). However, previous use of error propagation nets for speech processing has mainly been restricted to explicit presentation of the context (Elman and Zipser, 1987; Robinson, 1986) with some work using units with feedback links to themselves (Watrous, Shastri and Waibel, 1987). In a similar area, static error propagation nets have been used to perform image coding as well as conventional techniques (Cottrell, Munro and Zipser, 1986).

3.4.1 The Architecture of a General Coder

The coding principle used in this section is not restricted to coding speech data. The general problem is one of encoding the present input using past input context to form the transmitted signal, and decoding this signal using the context of the coded signals to regenerate the original input. Previous sections have shown that dynamic nets are able to represent context, so two dynamic nets in series form the architecture of the coder, as in figure 8.

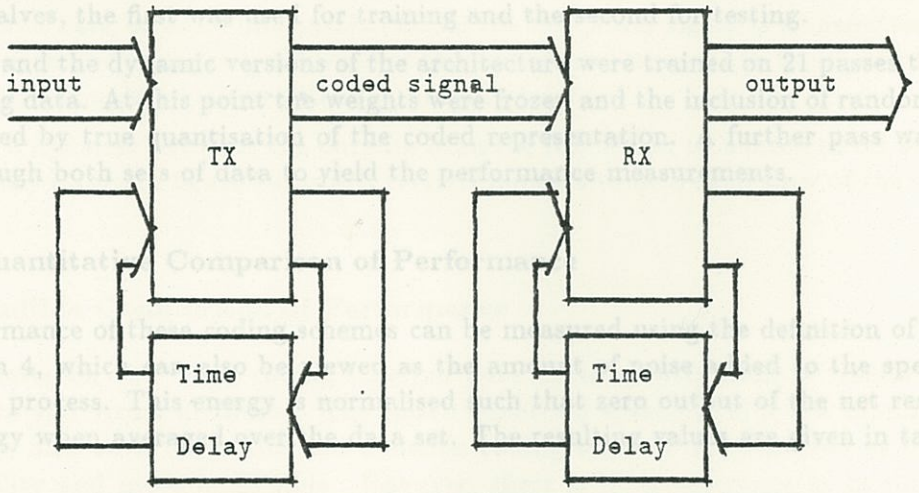


figure 8

This architecture may be specified by the number of input, state, hidden and transmission units. There are as many output units as input units and, in this application, both the transmitter and receiver have the same number of state and hidden units.

The input is combined with the internal state of the transmitter to form the coded signal, and then decoded by the receiver using its internal state. Training of the net involves the comparison of the input and output to form the error signal, which is then propagated back through past instantiations of the receiver and transmitter in the same way as a for a FIR dynamic net.

It is useful to introduce noise into the coded signal during the training to reduce the information capacity of the transmission line. This forces the dynamic nets to incorporate time information, without this constraint both nets can learn a simple transformation without any time dependence. The noise can be used to simulate quantisation of the coded signal so quantifying the transmission rate. Unfortunately, a straight implementation of quantisation violates the requirement of the activation function to be continuous, which is necessary to train the net. Instead quantisation to n levels may be simulated by adding a random value distributed uniformly in the range $+1/n$ to $-1/n$ to each of the channels in the coded signal.

3.4.2 Training of the Speech Coder

The chosen problem was to present a single sample of digitised speech to the input, code to a single value quantised to fifteen levels, and then to reconstruct the original speech at the output. Fifteen levels was chosen as the point where there is a marked loss in the intelligibility of the speech, so implementation of these coding schemes gives an audible improvement. Both nets had eight hidden units, with no state units for the static time independent case and four state units for the dynamic time dependent case. A context of the last four samples was used to train the dynamic net.

The data for this problem was 40 seconds of speech from a single male speaker, digitised to 12 bits at 10kHz and recorded in a laboratory environment. The speech was divided into two halves, the first was used for training and the second for testing.

The static and the dynamic versions of the architecture were trained on 21 passes through the training data. At this point the weights were frozen and the inclusion of random noise was replaced by true quantisation of the coded representation. A further pass was then made through both sets of data to yield the performance measurements.

3.4.3 Quantitative Comparison of Performance

The performance of these coding schemes can be measured using the definition of energy of equation 4, which can also be viewed as the amount of noise added to the speech by the coding process. This energy is normalised such that zero output of the net results in unity energy when averaged over the data set. The resulting values are given in table 2.

coding method	training data residual energy	testing data residual energy
linear quantiser	0.078	0.071
static net	0.070	0.075
dynamic net	0.057	0.056

table 2

Both the training and the testing data sets are large (about 200,000 samples), in comparison with the number of weights in the nets (54 in each static net and 146 in each dynamic net). Thus neither net could be expected to explicitly store individual values of input and output, but must make a generalisation to deal with the whole training set. Nevertheless,

there is some variability between the two sets and this is reflected in the different residual energies from the linear quantiser.

The static net is an improvement on the linear quantiser for the training data, but this does not generalise to the testing data. Inspection of the net shows that the quantisation levels are grouped more closely around the mean signal level and are sparsely spaced at the extremes. This reflects the distribution of values in the input stream.

The dynamic net performs significantly better than either the linear quantiser or the static net. Although an exact analysis of the net is not feasible, it is likely that the net forms a filter in the transmitter, and the inverse filter in the receiver. A larger net may form a source-filter model of speech, akin to Linear Predictive Coding.

3.4.4 Visual Comparison of Performance

A small section of the testing data is shown in figure 9. Figure 9a shows the original speech, a plosive burst and a vowel segment taken from the start of the word 'did'.

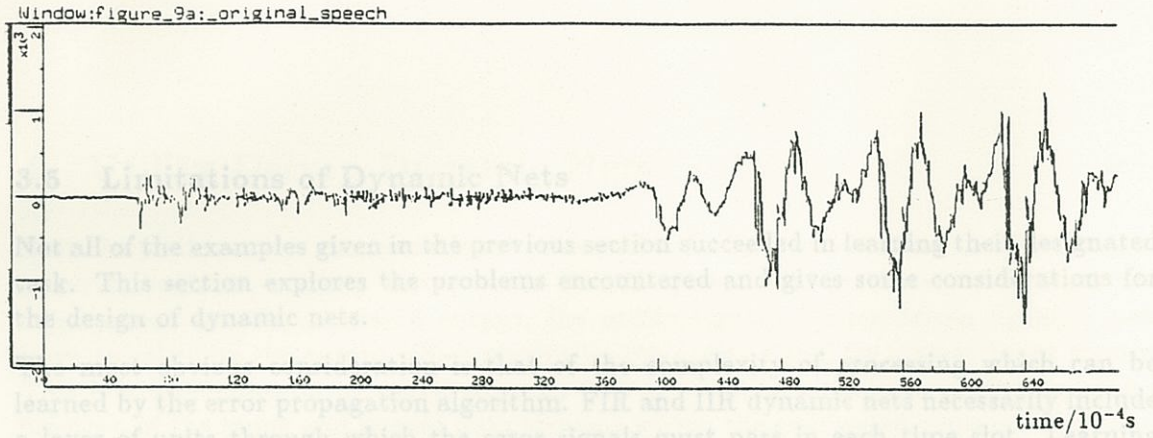
Figure 9b shows the original after processing by the linear quantiser. There is a marked loss of information, especially during the lower energy portion of the speech where most of the samples fall into the same quantisation band.

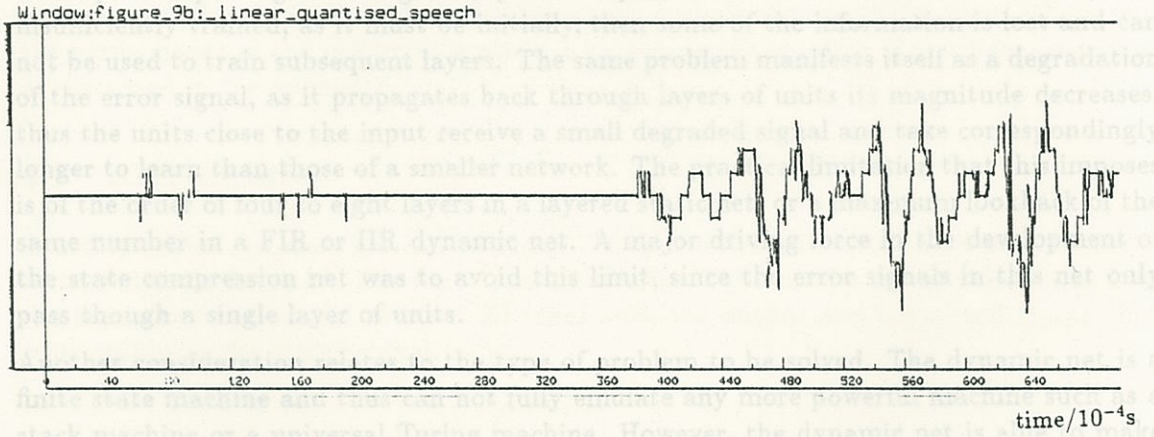
Processing with static nets gives an immediate improvement on the linear quantiser, as shown in figure 9c. The quantisation levels near the mean signal level are half as large as the equally spaced case, so more information about the low power signal is transmitted. This improvement in performance is achieved at the loss of resolution of the larger amplitudes, which in this example are also reduced in magnitude.

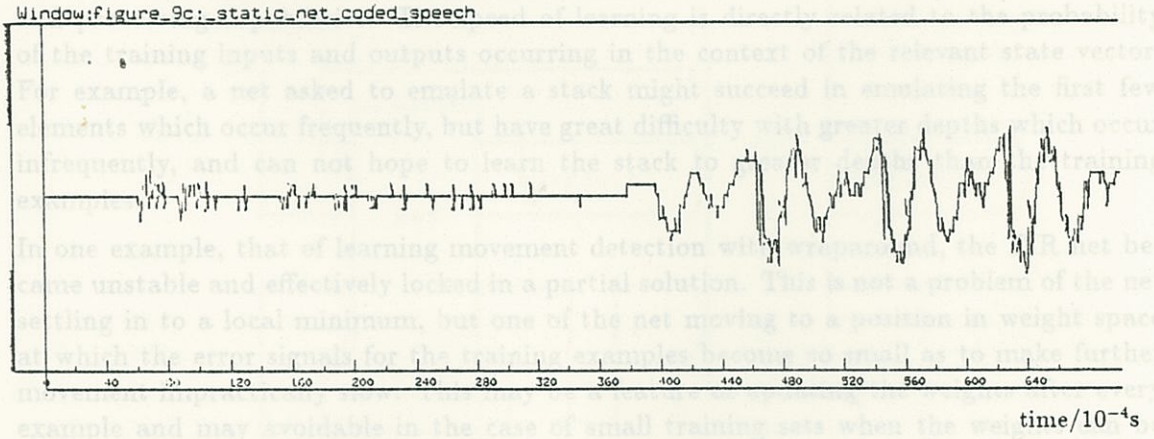
The use of a dynamic net frees the outputs from the restriction to a discrete set of levels. This is clearly shown in figure 9d, which visually appears to be a close copy of the original speech.

3.4.5 Audible Comparison of Performance

Listening tests on the three coding schemes gave different performance results to the quantitative and visual measures. The linear quantised speech is of a low intelligibility and quality, and both the static and dynamic coding schemes are reported as being of higher quality and more intelligible. However, there is little difference in intelligibility between the static and dynamic coders, which may be due to the ability of the ear to compensate for noisy environments. Dynamic nets do give higher speech quality than static nets, with less high frequency noise and a lower tone, which is more like the original speech. This improvement is consistent with the other performance results.


Window:figure_9a:_original_speech
x1e3
2
1
0
-1
-2
40
80
120
160
200
240
280
320
360
400
440
480
520
560
600
640
time/10^{-4}s


Window:figure_9b:_linear_quantised_speech
40
80
120
160
200
240
280
320
360
400
440
480
520
560
600
640
time/10^{-4}s


Window:figure_9c:_static_net_coded_speech
40
80
120
160
200
240
280
320
360
400
440
480
520
560
600
640
time/10^{-4}s

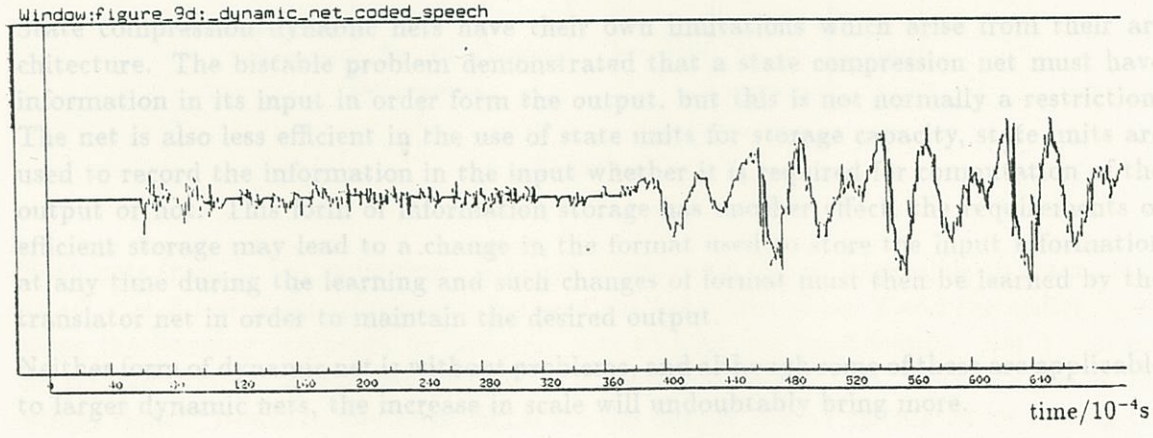

Window:figure_9d:_dynamic_net_coded_speech
40
80
120
160
200
240
280
320
360
400
440
480
520
560
600
640
time/10^{-4}s

figure 9

3.5 Limitations of Dynamic Nets

Not all of the examples given in the previous section succeeded in learning their designated task. This section explores the problems encountered and gives some considerations for the design of dynamic nets.

The most obvious consideration is that of the complexity of processing which can be learned by the error propagation algorithm. FIR and IIR dynamic nets necessarily include a layer of units through which the error signals must pass in each time slot. Learning to map an input signal though many such layers is a difficult task, for if one layer is insufficiently trained, as it must be initially, then some of the information is lost and can not be used to train subsequent layers. The same problem manifests itself as a degradation of the error signal, as it propagates back through layers of units its magnitude decreases, thus the units close to the input receive a small degraded signal and take correspondingly longer to learn than those of a smaller network. The practical limitation that this imposes is of the order of four to eight layers in a layered static net, or a maximum lookback of the same number in a FIR or IIR dynamic net. A major driving force in the development of the state compression net was to avoid this limit, since the error signals in this net only pass though a single layer of units.

Another consideration relates to the type of problem to be solved. The dynamic net is a finite state machine and thus can not fully emulate any more powerful machine such as a stack machine or a universal Turing machine. However, the dynamic net is able to make an approximate simulation of any machine within the restrictions of internal state space and processing capabilities. The speed of learning is directly related to the probability of the training inputs and outputs occurring in the context of the relevant state vector. For example, a net asked to emulate a stack might succeed in emulating the first few elements which occur frequently, but have great difficulty with greater depths which occur infrequently, and can not hope to learn the stack to greater depths than the training examples.

In one example, that of learning movement detection with wraparound, the FIR net became unstable and effectively locked in a partial solution. This is not a problem of the net settling in to a local minimum, but one of the net moving to a position in weight space at which the error signals for the training examples become so small as to make further movement impractically slow. This may be a feature of updating the weights after every example and may avoidable in the case of small training sets when the weights can be updated after all examples.

State compression dynamic nets have their own limitations which arise from their architecture. The bistable problem demonstrated that a state compression net must have information in its input in order form the output, but this is not normally a restriction. The net is also less efficient in the use of state units for storage capacity, state units are used to record the information in the input whether it is required for computation of the output or not. This form of information storage has another effect, the requirements of efficient storage may lead to a change in the format used to store the input information at any time during the learning and such changes of format must then be learned by the translator net in order to maintain the desired output.

Neither form of dynamic net is without problems, and although some of these are applicable to larger dynamic nets, the increase in scale will undoubtably bring more.

4 Utility Driven Dynamic Nets

Both the static net and the dynamic net presented so far have been trained on input/output pairs. In contrast, a utility driven net is trained by repreated presented of an input and a judgement on the calculated output, the utility signal. Utility driven dynamic nets learn to calculate the output which maximises the utility signal for a given input stream. This is a very powerful property, since all that is required is that the utility signal is a function of the input and output data streams within the considered context. Such nets have the potential to create complex internal models of the external world based on their own interrelationship with it. Applications may include the control of plant where only the performance of the system as a whole is known, or the building of intelligent machines as discussed in the appendix.

4.1 Architectures

A utility driven dynamic net can be formed from two dynamic nets and a possible architecture is given figure 10. The first dynamic net (net Y) computes the overall output from the input and the second net (net Z) takes both the output and input and learns their relationship to the utility signal.

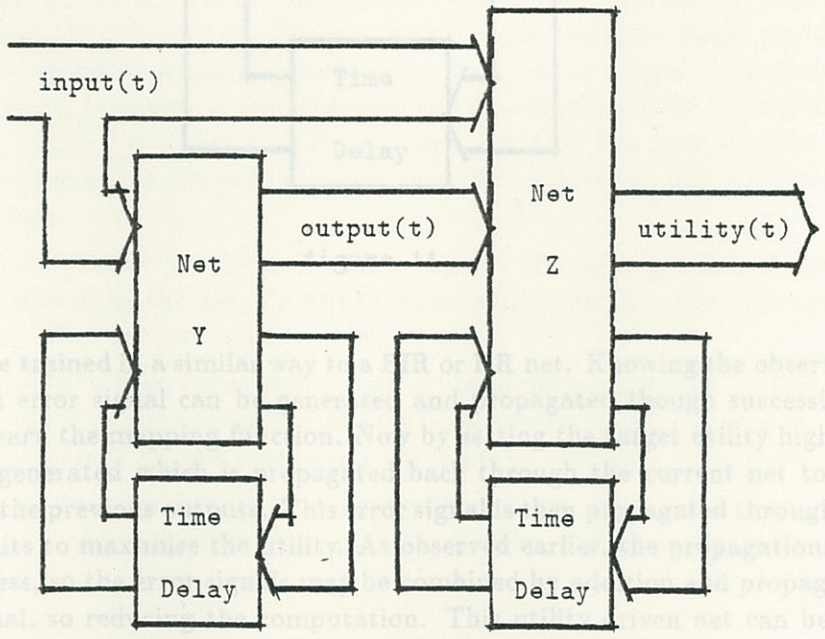


figure 10

Both nets train at the same time, although without net Z it is impossible to train net Y at all. Net Z learns the relationship between the behaviour of net Y and the utility signal using the learning procedure for dynamic nets without any modification. Net Y can not be trained directly as the desired output is only specified as that which maximises the utility signal. However, the error signal for the output can be calculated by setting the

output of net Z to its desired value, that is with the utility signal high, and propagating this error signal through net Z and subsequently though net Y. This is a new use for the error signal in back propagation networks, for it is simultaneously used to train a net and to generate the error signal for training another net.

In practice net Y and net Z may be combined to achieve a more compact net, as in figure 11.

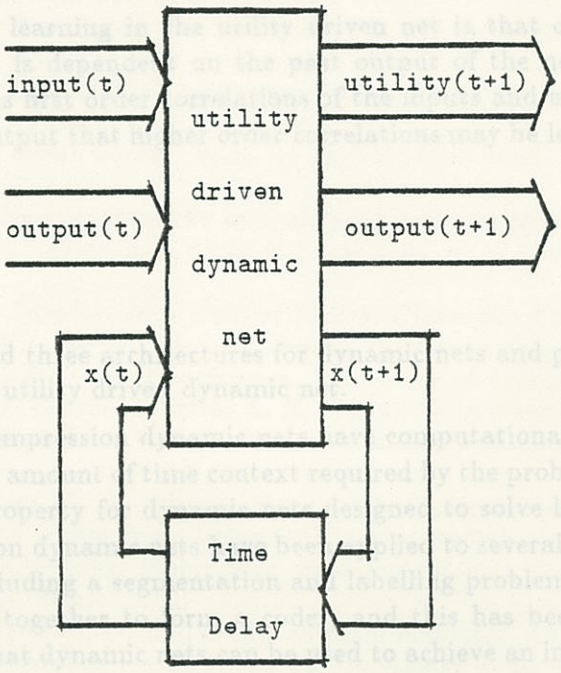


figure 11

This net may be trained in a similar way to a FIR or IIR net. Knowing the observed utility and output, an error signal can be generated and propagated though successive sets of state units to learn the mapping function. Now by setting the target utility high a second error signal is generated which is propagated back through the current net to form the error signal for the previous outputs. This error signal is then propagated through previous sets of state units to maximise the utility. As observed earlier, the propagations of errors is a linear process, so the error signals may be combined by addition and propagated back as a single signal, so reducing the computation. This utility driven net can be specified by the number of input, output, hidden and state units.

4.2 An Example Utility Driven Dynamic Net

A basic problem for a utility driven dynamic net is to copy a single input unit to an output unit with a time delay, the utility signal representing whether the copy operation was correct. Two hidden units and four state units were used. The utility signal was high if the sign of the previous output matched the sign of the preceding input and low otherwise.

There is no in built mechanism in the utility driven net to ensure that the net receives the full range of sensory inputs. In this example a learning rate, $\eta = 0.05$ or above tended to result in the output assuming a constant sign, the net then learned a utility function that was independent of the output, and so no maximisation of the utility signal was possible. If, however, η was set to 0.02 or below then several signs of output were observed whilst the utility function was learned, and so the utility function included the past output as a relevant input and the value of the utility could be maximised. With $\eta = 0.02$ this net can learn the target utility of 0.8 to an accuracy of ± 0.001 in 10,000 iterations.

The main problem with learning in the utility driven net is that of getting the utility function to learn that it is dependent on the past output of the net. It is hoped that in solving larger problems first order correlations of the inputs and utility signal will give sufficient variability of output that higher order correlations may be learned, although this remains to be shown.

5 Conclusion

This report has developed three architectures for dynamic nets and presented a new form of learning machine, the utility driven dynamic net.

The IIR and the state compression dynamic nets have computational requirements which are not dependent on the amount of time context required by the problem. This is believed to be a very desirable property for dynamic nets designed to solve larger problems. The FIR and state compression dynamic nets have been applied to several small problems with considerable success, including a segmentation and labelling problem. Two FIR dynamic nets have been coupled together to form a coder, and this has been applied to speech coding, demonstrating that dynamic nets can be used to achieve an improved performance in a real world task.

The utility driven dynamic net is demonstably capable of mapping a simple input stream onto an output stream by the use of a utility signal which describes the correctness of the current mapping. This net has considerable potential.

Several difficulties have been encountered, context information has been limited to a small number of previous time slots, instabilities have been observed, and learning from a utility signal is not guaranteed. Suggestions have been made to overcome these problems.

It is believed that dynamic nets, coders, and utility driven dynamic nets are capable of being scaled up to solve larger problems, and some effort will be put into demonstrating this in the near future.

One of the authors, A J Robinson, is supported by a maintenance grant from the U.K. Science and Engineering Research Council, and gratefully acknowledges this support.

References

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Journal of Cognitive Science*, **9**, 147–169.

Cottrell, G. W., Munro, P., and Zipser, D. (Febuary 1986). *Image Compression by Back Propagation: An Example of Existential Programming.* ICS Report 8702, Institute for Cognitive Science, University of California, San Diego.

Dennett, D. C. (1984). *Elbow Room: The varieties of free will worth wanting.* The Clarendon Press, Oxford.

Elman, J. L. and Zipser, D. (1987). *Learning the Hidden Structure of Speech.* ICS Report 8701, University of California, San Diego.

Hofstader, D. R. (1979). *Gödel, Escher, Bach: An eternal golden braid.* The Harvester Press, Hassocks, Sussex.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science U.S.A.*, **79**, 2554–2558.

Jacobs, O. L. R. (1974). *Introduction to Control Theory.* Clarendon Press, Oxford.

Jordan, M. I. (May 1986). *Serial Order: A Parallel Distributed Processing Approach.* ICS Report 8604, Institute for Cognitive Science, University of California, San Diego.

Kuffler, S. W., Nicholls, J. G., and Martin, A. R. (1984). *From Neuron to Brain: A Cellular Approach to the Function of the Nervous System.* Sinauer Associates Inc., Sunderland, MA, second edition.

Lindsay, P. H. and Norman, D. A. (1977). *Human Information Processing: An Introduction to Psychology.* Academic Press, Inc., Orlando, Florida, second edition.

Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA.

Pearlmutter, B. A. and Hinton, G. E. (1986). G-maximization: An unsupervised learning procedure for discovering regularities. In *Proceedings of the Conference on 'Neural Networks for Computing'*, American Institute of Physics.

Poggio, T. and Koch, C. (1987). Synapses that compute motion. *Scientific American*, May, 42–48.

Prager, R. W., Harrison, T. D., and Fallside, F. (1986). Boltzmann machines for speech recognition. *Computer Speech and Language*, **1**, 3–27.

Robinson, A. J. (1986). *Speech Recognition with Associative Networks.* M.Phil Computer Speech and Language Processing thesis , Cambridge University Engineering Department.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review*, **65**, 386–408.

Rosenblatt, F. (1962). *Principles of Neurodynamics.* Spartan, New York.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations.* (eds. D. E. Rumelhart and J. L. McClelland), Bradford Books/MIT Press, Cambridge, MA.

Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations.* MIT Press, Cambridge, MA.

Tank, D. W. and Hopfield, J. J. (1987). Neural computation by concentrating information in time. *Proceedings of the National Academy of Science U.S.A.*, **84**, 1896–1900.

Watrous, R. L., Shastri, L., and Waibel, A. H. (1987). Learned phonetic discrimination using connectionist networks. In *Proceedings of the European Conference on Speech Technology* (eds. J. Laver and M. A. Jack), CEP Consultants Ltd, Edinburgh.

A Appendix: Real and Artificial Intelligence

The aim of this appendix is to explore the relationship between a connectionist approach to artificially intelligent machines, and real intelligence as implemented by the human brain.

The specific example of a potentially intelligent machine, the utility driven dynamic net, has been investigated in the main body of the report. It is assumed that similar architectures are implementable for other connectionist learning algorithms such as G-Maximisation (Pearlmutter and Hinton, 1986), Boltzmann machines (Ackley, Hinton and Sejnowski, 1985) and Hopfield nets (Hopfield, 1982). Established theories in neurophysiology (Kuffler, Nicholls and Martin, 1984) and psychology (Lindsay and Norman, 1977) are examined in the light of an analogy of information processing in the utility driven dynamic net and in the human brain. This starts with an exploration of the analogy, and proceeds to discuss learning, memory, recognition, thought, consciousness and the implications of holistic processing. Thus it is hoped that the hypotheses presented in this section have a wide applicability, but it is to be stressed that, unlike the rest of this report, this appendix is subjective opinion without experimental verification.

A.1 Levels of Isomorphism

To compare two information processing systems it is useful to establish a level of isomorphism, (Hofstader, 1979) above which the two systems have the same behaviour, and below which the behavioural differences are unimportant. Firstly it must be established that it is possible for a level of isomorphism to exist. For this to be true we have to assume that both the brain and connectionist models have a finite processing and storage capacity, implemented by their hardware and with no concealed connections to the external world. This restricts both information processing systems to the class of finite state automata.

Traditional A.I. employs an isomorphism on the symbolic level and it is assumed that these symbols can be manipulated by formal rules. In contrast the connectionist approach employs a level of isomorphism on the signal processing level using many simple computation units which are massively interconnected.

Connectionist models do not aim to accurately models single neurons, but the units in a model are regarded as a neural like element. The intensity of signals in the brain are often measured as a firing rate, and the links between neurons are ascribed a synaptic strength, thus it is possible to draw an analogy between these quantities and the activations and weights in a connectionist network. However, the mathematically simple and uniform units used by connectionist models are in contrast to the complexity and diversity of types of neuron found in the brain. Thus a higher level of isomorphism must be used, and it is assumed that one or a few neurons can be modelled by a group of connectionist units.

Having established a level of isomorphism between the two information processing systems a comparison of computational complexity may be made. There are about 30,000 nerve fibres that connect the hair cells in the ear to the brain, about 800,000 that connect the eye to the brain and about 100,000,000,000 (10^{11}) neurons in the brain. Assuming that one or a few connectionist units are to be used to model each neuron, this yields a far larger machine than can currently be simulated. However, synthetic neural networks do stand a chance of showing interesting behaviour, the Aplysia (sea hare) contains just a

few thousand neurons while the leech contains only about four hundred. This is of the same order of complexity as current connectionist simulations, so we may expect to see similar behaviour.

A.2 Hardware Organisation

Some localisation of function occurs within the brain. Traditionally researchers have expected a greater degree of specificity than has been found. In contrast, the connectionist view is able to encompass a distributed representation of knowledge and computational power with no localisation of function. Localisation of function may be computationally efficient for a connectionist model, but is not necessarily an emergent property of the system.

The connectionist models of this report use highly interconnected units with nearly every possible connection made, but the brain uses sparsely interconnected neurons with only a few of the possible interconnections made. This difference in connectedness may account for the localisation of function, for example the restriction of neurons to a locally connected layered architecture close to the retina is a computationally sensible way performing a sequence of spatially local transformations on a visual input. These differences represent architectural constraints that must be correctly evaluated and built into the system as opposed to parameters which can be learned, and as such may represent an architectural barrier to the implementation of an intelligent connectionist machine.

As well as architectural constraints there may be an initialisation constraint. Some basic sensory and motor skills are built into mammals, but none are built into connectionist models. If a significant amount of the information required to produce intelligent behaviour is hereditary, then it is possible that evolutionary time periods as opposed to human lifetimes are necessary to train intelligent machines.

A.3 Learning

In 1898 Thorndyke proposed the law of effect: 'An action that leads to desirable outcome is likely to be repeated in similar circumstances'. This is good starting point for the quantification of utility driven connectionist nets, the desirability of the outcome corresponding to the utility signal. It is this maximisation of desirable outcomes which drives the learning, and it is assumed that a suitable definition for 'desirable' can be found. Possible candidates include a scale of pleasure/pain received by the sensors, or the degree of mismatch between the predicted sensory inputs and received sensory inputs, leading to pleasure seeking or knowledge seeking machines.

There are two possible extremes in the mechanisms of human learning. At one end the mechanism for learning may be built in from birth in the form of a large complicated program implemented on neural hardware. At the other extreme the mechanism for learning may itself be learned, so that the only function of the weight adaptation learning is to generate the neural program which allows for the observered learning. Thus there are possibly two levels of learning, the generation of a neural program and the implementation of that program. If these two levels do exist, then only the lower learning level may be discussed in the context of connectionist models, the higher level being more relevant to traditional A.I. It is an assumption of this report, and in the spirit of connectionism,

that the distinct levels do not exist and so the observed learning behaviour of humans and intelligent machines may be meaningfully discussed as a result of weight adaptation.

A.4 Memory

'There is reasonably good agreement that permanent storage of information takes place either through chemical or structural changes in the brain. There is little or no disagreement that the intermediate, ongoing activities of thought, conscious processes, and the immediate memories - sensory information store and short term memory - are mediated through electrical activity' (Lindsay and Norman, 1977, p421).

Similarly a dynamic connectionist network has two modes of information storage, as the transitory current activations of the units and the semi-permanent values of the weights. Input is combined with the current state to yield an output according the processing specified by the weights.

The first hypotheses is that the information stored in the neural activations is that of short term memory. There are far fewer nerve fibres entering the brain than there are neurons in the brain, by a factor of about 50,000, so there are easily enough neurons to account for short term memory if sensory input were to be stored as activations of neurons. Assuming a maximum firing rate of 1000Hz this represents about 50 seconds of raw data storage, and that is without using any form of data compression.

Using the connectionist model it is also possible to hypothesise about long term information storage. It is useful to split long term memory into two classes, semantic memory, that is memory about definitions which arises from repeated events, and episodic memory which pertains to single events. The hypotheses for semantic memory is the more fundamental of the two, as a characteristic feature of connectionist models is that they learn by repeated presentation of input-output pairs. Repeated presentation of input and utility signals train the net by modification of the format by which information is stored in the state space. For example, a net with little training may store a direct representation of the past few inputs in order to calculate the desired output. However, after more training the net will discover redundancies in the input which are not required for calculation of the output, so the format of the state space may be changed to represent the relevant information with more accuracy, or to represent more of the relevant information. This may be also be expressed by saying that the maximum information is stored in the state space if each state is visited with equal probability, and the machine changes the format of the state space to satisfy this condition. Thus repeated experiences result in a subset of the state space occurring with a higher probability, which is then compensated for by changing the weights to return the probability to its previous value. In this manner the information content of repeated short term memories are transferred to long term memory.

It is possible to extend the hypotheses to account for episodic memory. It is safe to assume that the facility to store single events is advantageous to the maximisation of the utility, and as a result this feature is likely to appear as an emergent property. The mechanism of this property is less easy to formulate. It is probable that the weights develop in such a way that important external events are maintained as patterns of activations for longer than less significant events, and the mechanism described for semantic memory can then act on these repeated states to form a long term memory. This links in with the well known memory aid of rehearsal to store short term memories in long term memory.

While activations can be changed at the same rate as the input stream, the format in which these activation are stored can only change at the same rate as the weights, or as the formations of long term memories. Thus it is possible to overload any one format for the activations by an unusual form of input, i.e. short term memory has a finite storage capacity limited by the form of long term memories. Of course both means of storage are finite, but this limited adaptability of short term memory may account for the observed 'hard limit', whilst the long term memories degrade at roughly their rate of formation which is much slower and gives the appearance of a 'soft limit'.

A.5 Recognition

Recognition is the task of matching partial information about an event with a stored internal representation. Recognition of sensory input may be considered as 'homing in' on a specific region in state space, with different regions corresponding to different events to be recognised. The direction of movement in state space is dependent on the sensory input and previous state vector, more information confirming or denying the recognition of an event by moving towards or away from the particular region that represents the recognition of that object.

This description of recognition makes it clear that the interpretation of sensory input depends on current internal state, perhaps this is most obvious in the multiple interpretations optical illusions, eg the 'Necker Cube' a two dimensional wire frame representation of a cube which can have two possible orientations when interpreted as a three dimensional cube. However, there is a wider applicability, what humans experience, 'see' or 'hear', is not what is presented as sensory input but is inevitably the result of the processing of this raw sensory data with an interpretation machine which has been trained on past experience.

This information processing strategy can be applied to deciding whether a task is a data driven or conceptually driven process. If the amount of information which is derived from immediate input is significantly larger that the information derived from the internal state, then the task is data driven, and if the reverse is true then the task is concept driven. However, it may well be impossible to quantify these amounts of information.

Differing recognition times for different events suggest that it may be necessary to proceed though many internal states before recognition is achieved. This is especially true when recall from long term memory is involved or when the recognised object does not fit a stereotype. Thus the differences in decision time between confirming 'a cat is a mammal' and 'a whale is a mammal' are those between initialising the internal state with the semantic form of the question, and arriving at the internal state which verbalises the answer. The different number of internal states that must be passed though reflects the fact that more information on 'whale' than 'cat' must be recalled before it can be confirmed that they match the stereotype.

A.6 Thought

The most startling conclusion from the connectionist assumptions is that man does not 'think'. This statement obviously needs some quantification, given a slowly varying internal structure then all decisions are based on a straight mapping of one internal state to

the next without any of the computationally expensive searches of traditional A.I. Thus the response to an input is pre-programmed, depending only on the internal state and the interconnection strengths. This assumption does not demean the processing capabilities of the brain, but presents an alternative model of processing using a finite state machine with a large number of states which is in contrast to conventional computers whose processor contains a finite state machine with far fewer internal states linked to external memory.

There are no levels of processing in the connectionists net, although the state vector could be considered as a 'blackboard' onto which sensory information is placed and used by other 'demons' which write their results onto the backboard. The information on the blackboard may be represented in a distributed form over the state vector and all the processing power of the demons distributed over the interconnections.

The 'amount of thought' required to complete a task can be related to the interaction of the input with the internal state. Reflex signals correspond to inputs which produce a determined output regardless of internal state, whereas a task which must be 'thought about' reallocates a large portion of the state space to the processing of the problem.

A.7 Consciousness

So far the discussion has only dealt with human information processing as observed from outside the system. However, each human is in a unique position of being able to comment on their own thoughts. Using introspection, humans can state that they are conscious and that they possess self-awareness. If this was merely a property observed in other humans, then it could be attributed to a learned ability to communicate some of their internal state variables to other humans, but because it is a property of yourself then this explanation seems inadequate.

For the brain or connectionist net to learn efficiently it must form an internal model of the external world. This model of the world must be centered on the machine itself, and must inevitably contain information about how the world will change as a result of the machines actions. This information about change includes the machines own actions, so the knowledge of ones own behaviour is explicitly stored and used in normal processing. Thus, self awareness may be seen as an emergent property of any machine complex enough to predict its own actions.

Dennett proposes an interesting way that this self awareness could develop into linear conscious thought (Dennett, 1984). A self-aware machine learns that it can solve problems by communicating with similar machines, then develops this into communication with itself, and finally dispenses with the external communication, 'talking to itself in an internal voice'.

A.8 Holistic Processing

Both information storage and information processing in a connectionist net are massively parallel distributed operations. Not only is it impossible to ask where a memory is located or where as task is computed, but it is impossible to know if they even exist unless the machine is examined externally. It is the conclusion of this section that a connectionist machine powerful enough to learn from real world experiences will develop a unique information storage and processing structure, interpretable only by exact simulation.