

2236608

DIGITAL NEURAL NETWORKS

This invention relates to digital neurons, and to neural networks comprising a plurality of such neurons, particularly but not exclusively for pattern recognition.

A neuron in this context is a circuit (realised with electrical or optical components) which receives a plurality of inputs and produces an output corresponding to a function (eg the sum) of the inputs each weighted by a respective weighting factor, derived during a training phase.

Attempts were made to inter-connect such neurons in layers, the output of one forming an input to another, so as to form a net. However, it was found that the effect of so doing was exactly equivalent to that of a single layer of neurons, unless a non linear compression stage was included between the layers.

In the implementation of neural networks, there is thus generally a requirement for non-linear activation function at the output of each neuron¹. This may take a number of forms, including the simple threshold function, but the most popular activation function is the sigmoid function, given by:-

$$y = \frac{1}{(1 + e^{-x})} \quad (1)$$

In the reported analogue integrated circuit implementations of neural nets, the non-linearity is usually implicit in one of the other neuron operations, for example in the analogue multiplier², or by allowing the analog summing amplifier to go into saturation and 'hit the rails'³. Thus the nature of the non-linear function is rather uncontrolled.

In digital Very Large Scale Integration (VLSI) implementations, by contrast, the activation function can be specified with arbitrary precision. A number of possibilities exist for evaluating the function of eqn. 1. It could be evaluated by summing a truncated series expansion, which is likely to be slow in terms of computation time, or by using a table look-up scheme in which the y value (output value) associated with each x (input value) value is stored in memory (eg Random Access Memory (RAM) or Read-only Memory (ROM)), and the x value is used to address the memory. If x is a 16 bit number and y is an 8 bit number, a simple look-up scheme would require 64 K Bytes of memory: this is clearly impractical, in terms of silicon area required, for each neuron of a VLSI neural network since such a network needs a very large number of neurons. One alternative would be to use a piecewise linear approximation scheme, in which the breakpoints are stored in memory, and table lookup is combined with linear interpolation.

According to the invention there is provided a neural processor comprising means for receiving an input signal and means for producing a digital output signal corresponding to a function of the input signal and a weight vector associated with the processor, further comprising means for applying a non-linear function to

said output signal to produce a digital compressed output signal wherein the function consists of a plurality of inclined linear segments with different slopes, the slope of each said segment being equal to a^n , where n is an integer and a is the base of the digital output signal.

This method of executing the activation function can be realised as a combination of a small number of simple logic gates (and shift stages) requiring no large storage table or repetitive calculation, and hence using less silicon area. Scaling to the different slopes is easy to achieve by left or right logical shifts since the slopes are powers of 2 (in binary arithmetic). It is inspired by the piecewise linear approximation used to implement A-law companding for Pulse Code Modulation (PCM) systems ⁴.

Other aspects of the invention are as claimed or described herein, with advantages which will be apparent from the following.

One embodiment of the invention will now be illustrated, by way of example, with reference to the accompanying drawings in which:

Figure 1 shows schematically a generalised digital neuron known in the art;

Figure 2a shows the general form of the sigmoid excitation function, and of a standard piecewise-linear A law curve;

Figure 2b shows an exemplary function curve according to a preferred embodiment of the invention, and the sigmoid excitation function;

Figure 3 shows schematically part of one example of a circuit for generating a non-linear function in a neuron according to the invention;

Figure 4 shows schematically a further part of that exemplary circuit;

Figure 5 shows part of the circuit of Figure 3 in greater detail;

Figure 6 shows one example of a circuit comprising an element of the circuit of Figure 5; and

Figure 7 shows a circuit, for use with that shown in Figures 3-6, for generating the gradient of the function shown in Figure 2b.

Referring to Figure 1, a digital neuron well known in the art comprises at least one input ($X_1, X_2, X_3, \dots, X_n$), means for scaling each input according to the value of a weight ($w_1, w_2, w_3, \dots, w_n$), and a processor P which produces a binary digital output Y which is a function (e.g. the sum) of these scaled or weighted inputs.

As discussed above, it is also in general necessary to provide means C for applying a non linear function to compress the range of the output Y, to give a compressed output I. This (together with other neuron outputs), is connected to form an input to a further neuron.

During a training phase, training data signal vectors are presented as input to the neuron, and the weights are altered (eg incremented) in dependence upon, amongst other things, the derivative of the (compressed) output by training means (not shown).

In our initial experiments a 13 segment piecewise-linear, A-law curve was used scaled such that values of neuron output x in the range -8 to $+8$ mapped to values of compressed output y in the range 0 to $+1$, to approximate the sigmoid function. Fig. 1 shows a plot of the scaled A-law curve, and for comparison the sigmoid function of eqn. 1. The nature of the piecewise linear A-law curve is explained in Reference 4, for example. However, simulations using this approximation indicated that it did not perform well when used to train Multi-layer Perceptron (MLP) networks using the Backpropagation Algorithm¹. This is because in the region around $x=0$, the first derivative (slope) of the A-law linear curve is much higher than that of the sigmoid function. As a result, when the sigmoid was replaced by the A-law curve, the training was unstable, and failed to converge. It could be made to converge by reducing the learning rate¹, but this resulted in very long learning times compared to simulations of the same network utilising the sigmoid function. It was found that the A-law curve could however be used in the recognition mode for nets trained (i.e. weight values derived) using the sigmoid function without serious degradation in the performance of the net.

A modified curve was therefore developed, similar to the A-law curve in that the gradient of each section can be expressed as a power of 2. This curve, shown in Fig. 2, has only 7 segments and is a better approximation to the sigmoid function, which is also shown in Fig. 2 for comparison. Note that at $x=0$ the modified curve and the sigmoid have the same gradient ($=0.25$). Table 1 compares the breakpoints of the A-law and the modified curves.

The breakpoints are conveniently at input values which are powers of 2; this enables the particular segment of the curve to which a given input value corresponds to be determined from higher order bits alone.

Simulation of the MLP with backpropagation using the modified piecewise linear approximation indicates that it gives comparable performance to simulations based on the true sigmoid, and is thus useful in trainable neurons and nets thereof.

Table 2 shows a possible truth table for the modified piecewise linear function corresponding to positive values of x , aimed at mapping a 16 bit 2's complement input value (I_0-I_{15}) in the range -8 to +8 to an 8 bit 2's complement output value (R_0-R_7) in the range 0 to 1. The negative half of the function corresponds in a obvious fashion. This could be used in a digital neural net system with 8 bit data word representation, which allows an additional 8 bits internal to each neuron for bit growth.

One possible hardware implementation of the truth table of Table 1 is shown in Fig. 3. This consists of a 'most significant 1' selector circuit 1, which takes the most significant bits $I_{11}-I_{14}$ of the neuron output as input, and outputs a 4 bit word $Y_{11}-Y_{14}$ containing a 1 at the position of the most significant 1 in the input, or else outputs 0 where there is no 1 in $I_{11}-I_{14}$. Signals $Y_{11}-Y_{14}$ of the neuron output are decoded to produce compressed output bits R_4-R_5 . Bit $R_6=1$ because the output is always greater than 0.5, in the positive quadrant of the function (shown in Fig. 1) and bit $R_7=0$ because the output is always positive. R_7 is therefore ignored (at this stage). Signals $Y_{12}-Y_{14}$ are also used to control a shifting circuit 2 comprising a bank of 2:1 multiplexer circuits 2a, 2b, 2c, which controllably shift bits of the

input word to provide the lower order compressed output bits R_0-R_3 . These multiplexers connect the upper right input to the output if the control signal is 0, and the lower right input to the output otherwise.

Fig. 4 shows one way in which the circuit of Fig. 3 can be generalised to cover both positive and negative values of x . If $I_{15}=1$ the neuron output is negative, and so the input is 2's complemented by a first complementing means 3a before being applied to the circuit of Fig. 2. The compressed output of the circuit of Fig. 2 (i.e. bits R_0-R_6) is also passed through second 2's complementing means 3b, which has the effect of mirroring the activation curve about the line $y=0.5$, and results in the correct value being output for negative inputs. For positive inputs, both 2's complement circuits 3a, 3b are bypassed. I_{15} is used as a control bit to switch 3a and 3b in or out. One's complementing could also be used, at the expense of a slight loss of accuracy, allowing simpler logic implementation of circuits 3a and 3b.

Referring to Figure 5, the 'select most significant 1' circuit takes $I_{11}-I_{14}$ as input, and outputs a bit word $Y_{11}-Y_{14}$ containing a 1 at the position of the most significant 1 in the input, or else outputs 0. This can be described by the following truth table:-

Table 3

I_{14}	I_{13}	I_{12}	I_{11}	Y_{14}	Y_{13}	Y_{12}	Y_{11}
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

(X = don't care)

This may be implemented in a number of ways. A modular implementation is shown in Figure 5. Each of the modules 1a, 1b, 1c, 1d shown in this figure has two inputs; an input I_i and an input P_{i+1} . P_{i+1} indicates whether there have been any more significant (previous) 1's input. i.e. $P_{i+1}=1$ if any $I_{j=1}$ where $j>i$. Each module has two outputs, Y_i and P_i . The truth table for each module is as follows:-

Table 4

P_{i+1}	I_i	P_i	Y_i
0	0	0	0
0	1	1	1
1	0	1	0
1	1	1	1

The required functions are thus; $P'_s = I'_s + P'_{s+1}$

$$(\overline{I'_s \cdot P'_{s+1}})$$

as shown in
Figure 6)

and; $Y'_s = \overline{I'_s} \cdot \overline{P'_{s+1}}$

$$(\overline{I'_s + P'_{s+1}})$$

as shown in
Figure 6)

This suggests one simple implementation of each module 1a, 1b, 1c, 1d using NOR, NAND and NOT gates shown in Figure 6. Other logically equivalent circuits are easily derived. As can be seen in Figure 5, P_{15} is set to 0, and P_{11} is a signal that is 0 if none of the inputs $I_{11}-I_{14}$ are equal to 1.

A further feature of the invention is that from outputs $Y_{11}-Y_{14}$ and P_{11} it is possible to derive a value for the gradient of the activation function, in other words the first derivative of the compressed output with very little additional circuit overhead. This is of course particularly useful during the training phase since it is required by the back-propagation algorithm.

The table below relates values of $Y_{11}-Y_{14}$ and P_{11} to the gradient of the curve, expressed as a 2's complement fractional binary number G_0-G_6 .

Table 5

Y_{14}	Y_{13}	Y_{12}	Y_{11}	P_{11}	G_6	G_5	G_4	G_3	G_2	G_1	G_0	
1	0	0	0	1	0	0	0	0	0	0	1	(1/64)
0	1	0	0	1	0	0	0	0	0	1	0	(1/32)
0	0	1	0	1	0	0	0	1	0	0	0	(1/8)
0	0	0	1	1	0	0	1	0	0	0	0	(1/4)
0	0	0	0	0	0	0	1	0	0	0	0	(1/4)

This gives the simple implementation shown in Figure 7. The gradient value output from this circuit (G_0-G_6) is valid regardless of the sign of the original input.

The circuit of Figs. 3 to 6 can be compactly and elegantly implemented in VLSI, using only simple logic functions which operate at extremely high rates compared to arithmetic processes, and occupy little chip area. Given the high operating speed, one potential application is in video image recognition, e.g. for a hybrid video coder (for example for use in a video phone). Other applications include speech conversion, industrial robot vision and natural language computing, and the principles behind it can be easily extended to other combinations of input and output wordlength.

It will be understood that the concept of the invention is applicable also to digital optical neural networks, and that it functions analogously with tri-state or other multi-level logic-state arithmetic; for an arithmetic system to base 'a', the slopes of linear segments are powers of 'a' and the segment breakpoints are powers of 'a'.

References

1. RUMELHART, D.E., and MCCLELLAND, J.L. (Eds): 'Parallel Distributed Processing' Vol. 1 (The MIT Press, Cambridge, Mass, 1986).
2. SCHWARTZ, D.B., and HOWARD, R. E.: 'A Programmable Analog Neural Network Chip' Proc. IEEE Custom Integrated Circuits Conference, 1988.
3. MEULLER, P. et Al. 'A General Purpose Analog Neural Computer' Proc. IEEE/INNS Int. Joint Conf. on Neural Networks, 1989.
4. SMITH, D.R. 'Digital Transmission Systems' (Van Nostrand Reinhold, 1985) pp. 78-88.

Tables

Table 1. Breakpoints of the A-law and modified piecewise linear curves.

Table 2. Piecewise linear activation function truth table (corresponding to Figure 2a) for positive values of input.

Table 3. Truth table for circuit of Figure 5.

Table 4. Truth table for circuit of Figure 6.

Table 5. Truth table for circuit of Figure 7.

BREAKPOINTS			
SCALED A-LAW		MODIFIED CURVE	
X	Y	X	Y
-8.0	0.0	-8.0	0.0
-4.0	0.0625	-4.0	0.0625
-2.0	0.125	-2.0	0.125
-1.0	0.1875	-1.0	0.25
-0.5	0.25	-	-
-0.25	0.3125	-	-
-0.125	0.375	-	-
0.125	0.625	-	-
0.25	0.6875	-	-
0.5	0.75	-	-
1.0	0.8125	1.0	0.75
2.0	0.875	2.0	0.875
4.0	0.9375	4.0	0.9375
8.0	1.0	8.0	1.0

TABLE 1

l_{15}	l_{14}	l_{13}	l_{12}	l_{11}	l_{10}	l_9	l_8	l_7	l_6	l_5	l_4	l_3	l_2	l_1	l_0	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
0	0	0	0	0	a	b	c	d	x	x	x	x	x	x	x	0	1	0	0	a	b	c	d
0	0	0	0	1	a	b	c	d	x	x	x	x	x	x	x	0	1	0	1	a	b	c	d
0	0	0	1	a	b	c	d	x	x	x	x	x	x	x	x	0	1	1	0	a	b	c	d
0	0	1	a	b	c	x	x	x	x	x	x	x	x	x	x	0	1	1	1	0	a	b	c
0	1	a	b	c	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	1	a	b	c

TABLE 2



CLAIMS

- 1 A neural processor comprising means for receiving an input signal and means for producing a digital output signal corresponding to a function of the input signal and a weight value associated with the processor, further comprising means for applying a non-linear function to said output signal to produce a digital compressed output signal wherein the function consists of a plurality of inclined linear segments with different slopes, the slope of each said segment being equal to a^n , where n is an integer and a is the base of the digital output signal.
- 2 A neural processor according to claim 1, in which the intersections of the said segments occur at digital output signal values of a^m , where m is an integer and a is the base of the digital output signal.
- 3 A neural processor according to claim 1 or claim 2, in which the said digital output is a binary number, the base a being 2.
- 4 A neural processor according to any preceding claim, in which the slope of the function around its centre approximates that of a sigmoid function.
- 5 A neural processor according to claim 4, in which the said slope of the function around its centre is 2^{-2} .
- 6 A neural processor according to claim 4 or claim 5, including means for altering the value of the said weight value in dependence upon the compressed output.



7 A neural processor according to any preceding claim, in which the non-linear function means comprises shifting means for logically shifting at least some lower order digits of said output signal, in dependence upon the value of the higher order digits of said output signal, the said digits thus shifted forming digits of the compressed output signal.

8 A neural processor according to claim 7, in which the non-linear function means further comprises a logic gate circuit connected to receive said high order digits and to provide an output for controlling said shifting means.

9 A neural processor according to claim 8, in which the logic gate circuit is for producing also the high order digits of said compressed output signal.

10 A neural processor according to claim 8 or claim 9, in which the logic gate circuit further includes means for generating a slope signal indicating the slope of the function corresponding to the said output signal.

11 A neural network comprising a plurality of neural processors according to any preceding claim connected so that the compressed output signals of some processors may form the input signals to others.

12 A pattern recognition device comprising a neural network according to claim 11.

13 A visual pattern recognition device comprising a network according to claim 11 arranged to receive signals derived from a video signal as inputs.

14 A neural processor substantially as described herein.
