WORKING NOTE AC78

Alan Creak

28 December 1990

SOME NOTES ON A PROPOSED INTERFACE DEVICE.

These notes are directed at extending my earlier work(1) (itself deriving from a much earlier – though not very informative – working note(2)) towards more practical implementation. They do not constitute a single topic; rather, they collect together a number of items of interest.

OBJECT.

Briefly, the aim is to construct a device which can be used as an interface unit between someone with physical disabilities and a computer. Its special features are :

- it can be trained to recognise a code designed to suit the physical abilities of the individual;
- the code may be composed of an arbitrary number of parallel channels, typically communicated through a set of switches matched to the person's needs.

Neural network techniques appear to be well suited to this aim, as they offer the flexibility, adaptability, and teachability which is needed to match an individuals needs and to adapt to comparatively slow changes in the person's abilities with time.

GENERAL PRINCIPLES OF INTERFACES.

Thakkar(3) presents four aims which should be addressed when designing devices for use by disabled people. They are :

- To maximise the available physical movements of the disabled user, with respect to reliability and durability, on any input or output device;
- To develop the cognitive and associated motor skills of the disabled users using the interfacing devices;
- To take into account human factors of physical components in the input devices used by the disabled user; and
- To keep the interfacing device affordable and flexible enough for future refinements.

Of these, the last is of direct concern, and indeed could be seen as a design principle of the proposed device. The third is addressed by our emphasis on the device's need to learn both a "language" appropriate to its user, and to adapt to changes in the language as the user becomes more proficient in its use. The first two, insofar as they are seen as essentially therapeutic aims, are not relevant to this device, though its ability in principle to exploit all possible channels of communication channels maximises the *utility* of the physical abilities, and it may well be profitable to design the "language" to stretch the user's skills. For the moment at least, though, these factors are out of the range of our consideration; the immediate aim is to demonstrate that the device will work.

MULTIPLE CHANNELS.

Clayton(4) reports a successful trial of a system using several switches to increase the communication rate of a disabled user. The system used six switches, and was used by activating two switches in sequence. The 36 codes were arbitrarily allocated to letters of the alphabet, digits, and control functions. From Clayton's paper : "Jamie learnt the codes (which were arbitrarily chosen) very quickly and uses the system very well indeed. He finds it a lot quicker than using his scanning system.". Clayton also presents a comment from Jamie himself, which contains these words : "Tell you something, it's quicker with the six ! ... I always say let's give it a go."

Another passing reference to parallel operation(12) appears in a description of a special multi-channel input interface, specifically designed to drive a graphics system. The interface incorporates a single switch, a keyboard, and a speech recognition system. "All three devices can be operated separately or in unison. This allows for the fit of several kinds of residual skill ..." While the authors clearly recognise the value of multiple input channels, this interface seems to be restricted to simple parallel operation, with no mention of its being trainable in any sense.

OTHER POSSIBILITIES.

The device is not restricted to switch input, and could in principle be used with a wide variety of signals from other sources. Here are a few examples, some perhaps more realistic than others.

A rather unusual input device which would fit well into this approach is an electroencephalograph (EEG). A report(5) on the possibility of detecting different forms of brain activity explicitly suggests that the technique could be useful for physically disabled people. Statistical techniques were used to analyse the EEG signals, and it was found that a selection of tasks – nothing, mathematics, two sorts of visualisation, verbal manipulations – could be distinguished from the EEG patterns. The authors do not consider that the mentally handicapped could make use of the method, on the grounds that the necessary control of mental processes might not be possible. I suspect that, if we could recognise brain activity connected with certain thoughts, then this restriction may be found invalid. Another study using generally similar techniques but relying on neural networks for discrimination(6) reports fair success in distinguishing EEG signals corresponding to the five vowel sounds, and to left, right, forward, and back hand movements.

On the other hand, superficially similar experiments based on electromyography (EMG) were markedly less successful(7), with imperfect resolution of EMG signals for only two quite different movements. It is interesting that the EMG experiments also demonstrated that different subjects produced quite different EMG patterns for the same motions; a trainable interface would again seem to be indicated, and it could be that a more flexible interface would learn to distinguish better than the statistical technique used(8). Neural networks have already been used to classify the signals acquired from muscle activity by electromyography(9)

An attractive, though possibly unexploited, approach would be to acquire input signals from a Kohonen network(10). The advantage of the Kohonen approach is that it gives a very versatile way of converting a wide variety of input signals into coordinate-based "feature maps", which seems to be very appropriate for our purposes.

WHAT SORT OF NETWORK ?

In exploratory work(1), I made enough progress to demonstrate the feasibility of the proposed device, but my network was of unusual type. The implementation was in effect a simulation of a hardware network composed of threshold logic units, without adjustable synaptic coefficients. In contrast with the conventional network, where the memory resides in the synaptic coefficients, all the memory of the implemented network was in the neurons' activities, most of which were binary. The connections between the neurons were engineered so that the network performed the required operations. On the credit side, the network was able

- to recognise correctly formed input signals by looking them up in a table;
- to adapt itself almost instantaneously to variations in input speed; and
- to cope with some malformation of the input signals.

As against these positive qualities, there were unsatisfactory features :

- the table of signals was in effect hard-wired, so that new or amended encodings could not be learnt without recoding the network;

- the design was too rigid to permit any more elaborate learning, such as would be needed to recognise idiosyncrasies of encoding.

To gain more benefit from the adaptability of neural networks, it would be helpful to combine the rapid "hard-wired" speed adjustment with a learning network which could learn new codes, and also to bypass the laborious multi-step analysis of the input signal, and in addition learn to identify any regular faulty, but recognisable, signal malformations.

These observations suggest some desirable qualities for the new implementation.

- It should be able to adapt essentially instantaneously to changes in input speed. It should have a component which could learn new codes, and make them available immediately.

- It should be able to look up a signal in its signal table if necessary (which will typically be the case for a newly defined sign). It should have a learning component, which will learn to correlate the input and output of the lookup operation, and to generate the output faster than the lookup phase. It should have an "error correcting" component which can make corrections backwards as soon as cues are available from levels of interpretation where there is redundancy, and adapt the earlier phases to reduce the rate of errors.

A system endowed with these abilities should behave rather like a human in its response to the input stream. These desirable features are suggested in the report of my preliminary investigation(1), though in my effort at implementation I explored only the first of the items in the list, relying on cheating for the table lookup.

It seems unlikely that a single homogeneous network will be able to meet all these requirements, and that some degree of separation of function in different components of the network will be necessary. For example, something like an ART(11) network seems appropriate for the second and third steps in the list. Indeed, there is a strongly Grossbergian feel to the problem, doubtless deriving from its rather specific and human-like requirements. For example, the switch from the learned response to the table lookup when the learned response fails is itself rather like the switch from one remembered pattern to another in ART until a satisfactory match is found.

SUGGESTED DEVELOPMENT PROGRAMME.

These suggestions are based on the observations above. Clearly, they may turn out to be inappropriate as we find out more about what we're doing, but they show the structure of the problem and give us somewhere to start.

I have assumed throughout that the input signal is Morse, as in my earlier investigation(1). The main advantage of Morse is that it's well defined and widely known, and it is the right sort of signal for our purposes. It is certainly not the only sort of signal we would want to use : the "other possibilities" above are largely concerned with analogue signals. Once the first stages were accomplished, it be useful to work in parallel on two sorts of signal – say, Morse and the sort of analogue coordinate signals one might get from a Kohonen network, as suggested earlier. (That doesn't mean we have to construct a Kohonen system : we can probably simulate it with a joystick, or something equally simple.)

1 : Design and construct (simulate) a network to learn from a list. This is the core of the rest of the system. It models the human ability to learn a cypher from a static table, and then to use the knowledge to decode incoming sequential signals. It should be adaptable to changes in the table, but need not cope with badly formed input signals or changes in input speed. (It should obviously be as fast as possible so that it can keep up with high input speeds, but this is a secondary consideration for the immediate problem.)

> I record some uncertainty about just how this stage should be managed. There are at least two possible ways to implement it :
>
> - Have the network learn all it can from the table before presenting the input signals. In human terms, learn all you can about perfect Morse before listening to any. Any change to the table would then require a relearning step, but that would not be a serious obstacle. With this approach, step 3 below would (I think) be unnecessary.
>
> - Have the network learn nothing until some real input is presented; then if it can't recognise the signal look it up in the table. No special procedure is needed if the table is extended. Step 3 is now essential.
>
> The second approach seems to me to be preferable on grounds of flexibility, but may be much harder. With the first method, we begin with a network that can at least recognise well formed input, which will be valuable in splitting the input stream into separate signs. The first method could also be implemented by well known back-propagation techniques. In the second method, the network should really learn how to look up a table rather than Morse as such, and it's harder to see how to do that.

2 : Think about the question of adapting to the input speed. Intuitively, it seems appropriate to think of this as a very early function in the information processing, which is how I implemented it in my own network. The result of this, though, is to present to later stages of the network a sequence of correctly decoded symbols, but still at a rate which depends on the input speed. In fact, therefore, the speed question must be tackled throughout the system.

3 : Add a learning-from-practice component which will improve its performance by practising. This models the human ability to improve with practice. It will presumably work by developing short-cut routes to solutions instead of laborious table lookup. It is something like an additional forward-acting network operating in parallel with the table lookup. The new learning component must continue to operate in conjunction with the lookup method, as the lookup technique will still be needed if the cypher is changed. An additional requirement is therefore that a successful response from the learning component must suppress the lookup. Notice that this may lead to interesting synchronisation problems if the input stream contains a mixture of recognised and unrecognised signals.

4 : The error-correcting component is the most difficult of the problems for the Morse network, though a cypher with more redundancy could be easier to handle. With Morse, no errors can be resolved until the language level is reached, when digram and trigram frequencies, or more elaborate measures based on language understanding, can be brought into play. However it is managed, it is likely to materialise in the form of feedback linkages which learn to use error indications developed late in the process to affect earlier interpretations.

Throughout the development, it will be important to keep in mind the needs of later stages of the network, and their demands on the earlier stages. For example, a Morse signal [dash, don't-know, dash] could represent either K or O; the error-correcting component is likely to be more helped by a representation which preserves the ambiguity than by one which arbitrarily decides on one of the possibilities. A data representation convention which can represent uncertainty is likely to be essential.

TESTING AND EVALUATION.

The ultimate test for the completed device will be its utility in increasing the communications rate for a disabled person, but that criterion isn't very helpful for day-to-day work, and is in any case unlikely to be satisfactory for quantitative assessment. It is therefore desirable to establish some sort of benchmark against which we can assess the performance of our implementation.

We can assert a few general propositions about a benchmark. It should be *reproducible*, so that we can repeat tests on consecutive versions of an implementation to measure improvement (or otherwise !); it should be *representative* of the conditions expected in the practical system, and not restricted to special cases; and at the same time it should be *explicit* for specific difficulties so that we can assess the value of different approaches on specific areas of difficulty. In addition, the benchmark should be designed so that it can be used to assess the system's effectiveness in each of the areas in which it is expected to function. For our system, a good benchmark would be a reproducible set of input signals of the sort which the final device will need to decode, chosen to reflect the different features to be tested.

For example, if the list of features above is taken as the set of properties to be implemented, we might suggest a benchmark along these lines :

- The basis of the test will be a selected set of sequences of input signals chosen to represent typical and specially taxing cases, and sufficiently broad in scope to permit reasonable statistical behaviour. These will be used together with a number of parameter sets which determine how the sequences will be presented, so that the content of the sequence and its transmission characteristics can be handled independently. The characteristics controlled by the parameters would include those suggested by the items in the list.

- To check the response to changes in input speed, both the speed and the rate of change of speed should be controllable. We would wish to measure the response to sequences presented at different constant speeds, investigating both the steady-state performance and the initial transient response. In addition, we need measurements on examples in which the speed changes through the sequence, in some cases steadily and in others abruptly.

- To check the response to new codes, there must be a way to present such new definitions within the sequences. If the usual methods of presenting new codes can be performed through normal input sequences, that's easy; if not (as seems more likely), some special provision must be incorporated in the testing procedures. The ability to look up a signal in a signal table is likely to be the first function implemented in the network. Later on, as (if !) the more elaborate developments eventuate, it may be necessary to incorporate some sort of monitor which will report on the route by which the output signals were generated – and, doubtless, other matters regarding the internal functioning of the system. The monitor suggested will also be useful in testing the learning component. Repeated presentation of the same sequence should generate output signals faster as the learnt response takes over from the systematic decoding.

- To test the error correcting performance, it will be necessary to design specific input sequences to fit the tests required. For example, the performance on random letter sequences could be compared with the performance on stochastically generated sequences with ideal statistical behaviour, and also with real text.

Such a benchmark could be implemented in a number of ways. The most stringent, the most realistic, and the hardest to implement is an emulation of the behaviour of a real device generated in a machine separate from the processor which implements the network. For regular use, though, we can probably derive much useful information from a simulated input – perhaps already digitised and quantised in some way – which would replace the "real" input procedure.

REFERENCES.

1 : G.A. Creak : *The adaptive peripheral : a teachable interface based on a neural network,* Auckland Computer Science Report #47, August 1990.

2 : G.A. Creak : *An intelligent interface for the disabled,* unpublished Working Note AC45, 1985.

3 : U. Thakkar : "Ethics in the design of human-computer interfaces for the disabled", *Sigcaph Newslette*r **#42**, 1 (June 1990).

4 : C. Clayton : "Building a system for Jamie Todd", *ISAAC UK Newsletter* **#5**, 2 (March 1989).

5 : Z.A. Keirn, J.I. Aunon : "Man-machine communications through brain-wave processing", *IEEE Eng. in Med. and Biol.* **9#1**, 55 (March 1990).

6 : A. Hiraiwa, K. Shimohara, Y. Tokunaga : "EEG topography recognition through neural networks", *IEEE Eng. in Med. and Biol.* **9#3**, 39 (September 1990).

7 : G. Hefftner, G.G. Jaros : "The electromyogram as a control signal for functional neuromuscular stimulation – Part II : Practical demonstration of the EMG signature discrimination system", *IEEE Trans. Biomed. Eng.* **35**, 238 (1988).

8 : G. Hefftner, W. Zucchini, G.G. Jaros : "The electromyogram as a control signal for functional neuromuscular stimulation – Part I : Autoregressive modeling as a means of EMG signature discrimination", *IEEE Trans. Biomed. Eng.* **35**, 230 (1988).

9 : M.F. Kelly, P.A. Parker, R.N. Scott : "Myoelectric signal analysis using neural networks", *IEEE Eng. in Med. and Biol.* **9#1**, 61 (March, 1990).

10 : for example, T. Kohonen : "The 'Neural' phonetic typewriter", *IEEE Computer* **21#3**, 11 (March 1988).

11 : for example, G.A. Carpenter, S. Grossberg : "The ART of adaptive pattern recognition by a self-organising neural network", *IEEE Computer* **21#3**, 77 (March 1988).

12 : A. Tronconi, M. Billi, A. Boscaleri, P. Graziani, C. Susini : "Graphics with special interfaces in Italy", *Communication Outlook* **10#3**, 6 (Winter 1989).