VLSI IMPLEMENTATIONS OF ELECTRONIC NEURAL NETWORKS: AN EXAMPLE IN CHARACTER RECOGNITION

L. D. Jackel, B. Boser, H. P. Graf, J. S. Denker,
Y. Le Cun, D. Henderson, O. Matan, R. E. Howard
AT&T Bell Laboratories, Holmdel NJ 07733, and
H. S. Baird, AT&T Bell Laboratories, Murray Hill NJ 07974

By their very nature, neural-net chips are application specific circuits. Because the lead-time for chip design, fabrication, and testing usually exceeds a full year, and because neural-net algorithm development proceeds at a faster pace, the circuit designer must face the problem of creating a circuit that will be useful for a network application that is not even conceived when circuit design begins. If we could cram unlimited circuitry on a chip, we could build enormous fully-connected hardware networks; particular network designs could be implemented by pruning away the superfluous connections. Unfortunately, a fully-connected network structure that includes most *useful* networks as subsets would be two or three orders of magnitude larger than anything we can build today, and it would be so wasteful of area and power that such an approach would probably never make sense.

We have, however, found a more economical approach that is far more promising. Through a series of experiments in pattern recognition using neural-net algorithms, we identified a large class of applications where theoretical considerations that promote high-accuracy classification result in constrained network architectures. These constrained nets can map nicely onto appropriately designed hardware. Here, we discuss the concepts we learned from our pattern recognition experiments, we show how they can be applied to chip design and we describe a new neural-net chip.

Introduction

In this paper we will use the example of Optical Character Recognition (OCR) to illustrate neural-network architecture principles, and in particular we will show how these principles can influence the design of neural-net chips. This paper is not intended as a review of current neural-net chip technology [1].

The most commonly used neural networks are called "feed-forward, layered networks." Each layer in such a network has an array of inputs from the previous layer and sends its output array to the succeeding layer. For OCR, the input to the first layer is typically a pixel-map of a character image. The output array usually has one component for each category, with the value of a particular component expressing a measure of the likelihood that the input image represents a character of the corresponding category.

A major advantage of neural-network methods is that they allow learning from examples. In the case of pattern recognition, we wish to learn a function that takes an image as its input and produces a category label as its output. The untrained network is a parameterized family of functions, where the parameters are the weights used in the weighted sums. Training consists of selecting a suitable function from this family, by searching for appropriate values of the weights. The technique known as Backward Error Propagation (or simply backprop)[2] is an efficient method for carrying out this search.

An added bonus of the neural-net approach is that the predominant arithmetic calculations are multiply-add steps, most of which can be performed in parallel, either on standard Digital Signal Processors (DSPs), which are fully programmable, or on special purpose neural-net chips. DSPs can perform multiply-adds at rates of about 10 million per second which is faster than most commonly used microprocessors. Neural-net chips can do the same calculation (though usually with fewer bits of resolution) at speeds of over 1 billion per second.

Despite the speed advantage of custom chips, neural-net chip designers have been plagued by the following dilemma: with today's VLSI technology, it is impossible to design chips that are flexible enough to encompass all possible network architectures and, at the same time, contain enough resources to accommodate the large networks needed for many real-world applications. Designers typically guess that the ultimate users of the chips will want several fully-connected layers of neurons with as many neurons and connections as will fit in the system. This approach is often unworkable because the number of connections between fully interconnected layers of neurons grows as the product of the number of neurons in the layers. Even in a small image-processing problem, the input field is at least 16 x 16. If the number of neurons in the next layer of the net is the same size as the input field (it is usually larger), then (16x16) x (16x16) = 64,000 connections are needed. The largest neural-net chip made so far [3] has 32,000 1-bit connections, so it falls short of the needs of this apparently minimal network. In practice, single-bit connections are likely to be insufficient for our needs, and several 1-bit connections would have to be combined to provide composite connections with the required analog depth. Even though VLSI technology is advancing, this type of completely parallel, one-chip network will not be readily available until near the end of this decade.

The prospects for the near-term availability of really useful hardware are not as bleak as the previous paragraph might suggest. Through our work on various character-recognition systems, we have discovered that at least for this class of problem, constraints imposed on the network to improve classification accuracy result in an architecture having limited connectivity and replicated weights, leading to an ideal structure for time-multiplexing the hardware.

Character Recognition and Feature Extraction

In optical recognition, a character must be identified from the pixels in its image. Depending on the nature of the image, this task ranges from extremely difficult to fairly straightforward. If the image includes only cleanly printed, fixed-font symbols, simple template matching does an excellent job of classification. When multiple fonts are used, or when the image is noisy or distorted, template matching is not as successful, and more sophisticated methods, usually including feature extraction, must be used. The problem becomes even more difficult with hand-printed characters because of the wide variability in writing styles.

We can think of the neural-net character recognizer as implementing a mapping from an input space whose dimension is the size of the image pixel-array (say 16x16 = 256) to a 10-dimensional output space (one output for each digit category). If the mapping were regular, points near each other in input space (in a simple metric such as Euclidean distance) would be mapped to the same point in output space (same category); in such a case,

90CH2930-6/90/0000-0320$01.00 © 1990 IEEE

simple techniques, such as learning with a small, fully-connected backpropagation net, would do the job. Alas, the mapping required for this task is much nastier: even characters that appear essentially the same to a human are often far apart in a simple metric. The distance between character images within a class is often greater than distances between character images in different classes. In this case, backpropagation through an unstructured net cannot provide acceptable results. This is because in such a general network, an enormous amount of training data would be needed to determine the huge number of free parameters in the network [4]. With limited data, undetermined free parameters make it easy to "learn" the training data, but result in poor "generalization" when the network has to classify characters not in the training set. Even if sufficient data could be found, the learning time would be prohibitively long.

The "classical" solution of this problem is to change the input representation of the input data to bring elements of the same category closer to each other than to members of different categories. Feature extraction is a well-known change-of-representation technique.

Recently, Le Cun et al [5], working with a database of hand and machine printed digits, developed a series of networks that could be *trained* to develop their own feature extraction kernels. They designed a network with a limited connectivity where the first few layers performed multiple convolutions. Groups of weights in the net were constrained to take the form of convolution kernels, but their values were *not* fixed. This network had many fewer connections than a fully-connected net with comparable computing power. The constraints on the network drastically reduced the number of free parameters, thereby improving generalization. The classification accuracy was state-of-the art: when tested on hand-printed zipcode digits normalized to 20 x 20 pixel images, 7% of the digits had to be rejected to keep substitution errors below 1%. (While this may seem like a high reject rate, it is actually quite good considering the poor quality of the test images.) Although this network had 100,000 connections it had only 2,500 free parameters. With this number of connections, the system could process one digit per second on a Sun 3 workstation. Using an AT&T DSP32C Digital Signal Processor in conjunction with either the Sun or a 386PC, the rate increased to 10 digits per second. Evaluation of the network took only 30msec; most of the computing time was taken by image scaling.

Recognition Network Design

From the proceeding example, and from other recognition experiments [6,7], certain network design principles emerge. While we cannot be certain that they *must* be common to all machine perception systems, it is likely that they will be important.

Here is what we find: the early layers of the network have limited receptive fields; that is, each neuron in these layers takes its input from a small, compact region of the previous layer. Moreover, the units (or neurons) in these layers can be grouped so that their outputs form maps whose dimensionality is the same as the input field (obviously 2D for vision problems). Most importantly, units within a map perform the same operation as their neighbors, but operating on a shifted input. Explicitly, it seems that the right thing to do is to take convolutions followed by a non-linear function on each output component.

We find that high resolution is important for defining the types of features that will be detected, but rather less precision is required for recording *where* a given feature occurred. In turn, progressively higher-level features require progressively less precision in their location. Therefore the outputs of the feature maps tend to have less spatial resolution than their inputs.

Biology seems to have come up with the same principles in animal vision systems, where numerous feature maps are formed for the visual field; this gives us additional confidence in our approach. But biological neurons are much "cheaper" than silicon neurons; our brains cram 10^{15} synapses into a two liter volume. Even if we knew how to hook them up, that many silicon connections, stacked as chips 1 cm apart, would fill a room 10 meters square and a few meters high.

Silicon neurons, however, are perhaps 100,000 times faster than biological neurons. Clearly, systems using electronic neurons would do well to use time-multiplexing to compensate for their limited parallelism by taking advantage of their high speed. This tradeoff will not work if all the connections of a network have different, random values since we would have to load every neuron's unique set of input connection weights onto the chip, creating an enormous I/O load on the system. The fortunate result of our character recognition experiments is that in order to learn effectively from examples or even to do feature extraction with fixed, hand-crafted kernels, we need a highly regular, locally connected network with repetitive use of the same weights. This is precisely the kind of network that lends itself to multiplexing.

Another feature of our recognition networks that aids multiplexed processing is that even with the coarse-blocking of the feature maps, successive layers of feature extraction still just require input data whose origin can be traced to a restricted region of the image input field. This means that if we provide some modest on-chip storage, we can serially process portions of the image several layers deep through the net before requiring new input data, significantly reducing I/O, and easing a potential bottleneck.

A Neural-Net Chip for Machine Vision

In this section we describe a second generation neural-net chip [3], designed by H. P. Graf that includes support circuitry that makes it efficient for image processing. The chip, which combines analog and digital processing, is reconfigurable, so that the number of "neurons" can be varied as well as the number of connections per neuron. The 32,000 programmable 1-bit connections on the chip can be combined to form fewer connections with greater analog depth.

The chip which measures 4.5 mm by 7.0 mm, was fabricated by AT&T in 0.9 micron CMOS. It can evaluate all 32,000 connections in parallel (with 4% accuracy on the weighted sums) every 100nsec, giving a rating of 300 billion connections/sec. The connections each take up a cell 23 microns by 17 microns and contain 9 transistors. The chip also contains thresholding circuitry, multipliers, switches to set the network architecture, and decoders, as well as output shift registers to control the data flow. The shift registers can also be used to store intermediate results, passing the output of one neuron to the input of another.

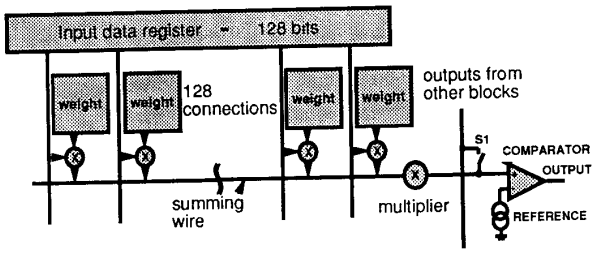

Figure 1. Basic building block "neuron."

The chip's basic building block "neuron" is shown in Figure 1. An array of 128 SRAM cells store the 1-bit connection weights. Each connection is evaluated locally using a digital NXOR. Outputs from the NXORs are connected to current sources to permit efficient analog summing. Using current mirrors, the summed current is scaled by a multiplier and can be added to the currents of other building blocks. The resulting current is converted to a voltage that can be processed by a comparator with a programmable reference threshold.

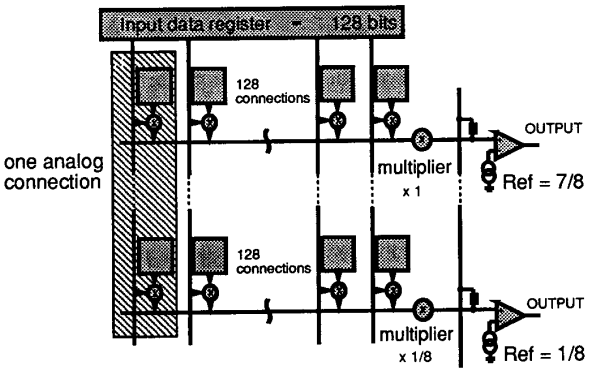

Figure 2. Combining blocks to form neurons with analog depth.

Greater effective analog depth in the weights can be attained by combining several blocks of neurons as shown in Figure 2. The same input data is passed to all the blocks of neurons. A column of 1-bit weight SRAM cells stores a binary-coded weight value. The multipliers are set to scale as powers of 2, and the outputs of the blocks are combined to form the appropriately weighted sum. Each block also provides a comparator to sample the common summing wire; by using different reference values for each comparator, a flash A/D conversion can be done, including a non-linear sampling if desired.

The blocks can also be combined as shown in Figure 3 to provide additional inputs. In this case the multipliers are set to the same value and different input data is sent to the different blocks. Up to 8 blocks can be joined either in the manner of Figure 2 or Figure 3 or in combination of both schemes. For example, 8 neurons could be used to form a neuron with 256 inputs and 4-bit weights.

Using these methods, the first layer of the OCR net was processed by the chip. Efforts are now continuing to map most of the recognizer network onto the chip.

Conclusions

We have found that different approaches to image recognition often lead to neural-net architectures that have limited connectivity and repeated use of the same set of weights. This architecture is ideal for time-multiplexing (a combined parallel-serial processing) on hardware systems that would be too small to evaluate the entire network in parallel. To make this process efficient, a chip needs to have shift-registers to format the input data, and additional registers to store intermediate results. Within this framework, it is possible to design chips that have broad utility, large connection capacity, and high speed. This was demonstrated by a new chip with 32,000 reconfigurable connections.

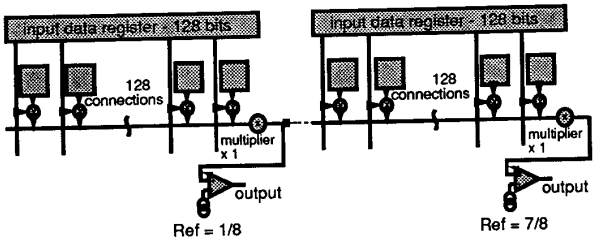

Figure 3. Combining blocks to form neurons with additional inputs.

Acknowledgement

We thank the U.S. Postal Service for providing us with a database of handwritten digits.

References

[1] For a snapshot of the state-of-the-art in neural-net chips see the the section on "Electrical Neurocomputers", in the proceedings of the IJCNN, 1990, San Diego, IEEE Catalog Number 90CH2879-5.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Back-Propagating Errors", Nature, **323,** pp 533-536 (1986).

[3] H. P. Graf and D. Henderson, "A reconfigurable CMOS neural network", Digest of Technical Papers, IEEE International Solid-State Circuits Conference, 1990 pp. 144, 145, 285.

[4] J. S. Denker, D. Schwartz, B. Wittner, S. A. Solla, R. Howard, L. Jackel, and J. Hopfield, "Large Automatic Learning, Rule Extraction and Generalization", Complex Systems, **1,** pp. 877-922 (1987).

[5] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and H. S. Baird, "Handwritten Zip Code Recognition with Multilayer Networks", Proceedings of 10th International Conference on Pattern Recognition, (Atlantic City, NJ USA), Volume 2, pp. 35-40, IEEE Computer Society Press, Los Alamitos, CA (1990).

[6] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I Guyon, "Neural Network Recognizer for Hand-Written Zip Code Digits", in *Advances in Neural Information Processing Systems 1,* David S. Touretzky, ed., Morgan Kaufmann, San Mateo, CA, pp. 323-331 (1989).

[7] I. Guyon, P. Albrecht, Y. Le Cun, J. Denker, W. Hubbard, "A Time Delay Neural Network Character Recognizer for a Touch Terminal", Proceedings INNC 90 Paris, Kluwer Academic Publishers, pp. 42-45 (1990).