HARDWARE REQUIREMENTS FOR NEURAL-NET OPTICAL CHARACTER RECOGNITION

L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, Y. Le Cun
I. Guyon, D. Henderson, R. E. Howard, W. Hubbard, and S. A. Solla
AT&T Bell Laboratories, Room 4D-433
Holmdel, NJ 07733

Abstract

Through a series of experiments in optical character recognition, an understanding is beginning to emerge of the general nature of the hardware required. Rather than the fully-connected layered neural nets conceived by most hardware researchers, many machine perception tasks require local connectivity and repeated weight patterns between layers to support computing of convolutions. No current-day hardware is available to evaluate in parallel all the connections in a character recognition system. Fortunately, the repetitive nature of the convolution operation makes time-division multiplexing of the hardware possible and even efficient. To avoid I/O bottlenecks, the hardware must contain substantial input data buffers and shift registers. I/O requirements are further relaxed if several layers of the net are processed in a pipelined fashion without recourse to external storage. This paper will discuss hardware architectures for character recognition and will outline choices for possible circuits. An advanced (and working) reconfigurable neural-net chip, that mixes analog and digital processing, will be described.

Introduction

Neural-net hardware designers have been plagued by the following dilemma: with currently available electronics technology, it is impossible to design hardware that is flexible enough to encompass all possible network architectures and, at the same time, have enough resources to accommodate the large networks needed for many real-world applications. Designers typically guess that the ultimate users of the hardware will want several fully-connected layers of neurons with as many neurons and connections as will fit in the system. This is clearly a losing proposition 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 [1] 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, pushing the arrival of a completely parallel, one-chip network toward the beginning of the next millennium.

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 systems to recognize characters we have discovered, that at least for this class of problem, constraints imposed on the network to improve classification accuracy result in an architecture that has limited connectivity and replicated weights, leading to an ideal structure for time-multiplexing the hardware. We will describe several architectures for character recognition, stressing the characteristics they have in common. We will then discuss the hardware options available to implement these architectures, including how the most compute-intensive parts of the network can be mapped onto a new, (and fully-tested) neural net chip.

The Character Recognition Task

In optical recognition, a character must be identified from its image pixels. Depending on the nature of the image, this task ranges from being extremely difficult to being 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, template matching is not as successful, and more sophisticated methods, usually including feature extraction, must be used. Nevertheless, it is still possible to read a full page of poorly printed text with variable fonts and make only a few errors. Many of these can be eliminated using contextual information from syntax and semantics [2].

The problem becomes much more difficult with hand-printed characters. For this problem simple template matching does not work because of the variability of writing styles. When there are breaks within a character, or when characters are touching, the task becomes even more difficult. Here, the job of segmenting the image to determine which pixels belong to which characters becomes both the time and accuracy rate-limiting processes. Currently, to read 5-digit hand-printed zipcodes buried in the confusion of all the writing on a typical envelope, the best systems must reject about 60% of the images to insure that the zipcodes in the remaining images are read correctly with 99% confidence. Machine reading of un-formatted, hand-*written* script is a dream we are unlikely to see come true since many of us can't even read our own writing.

Feature Extraction

We can think of designing a neural net for the recognition task as equivalent to creating a mapping from an input space whose dimension is the size of the image bit-array (say 16x16 = 256) to a 10-dimensional output space. If the mapping were regular, so that points near each other in input space (in a simple metric such as Euclidean distance) also mapped to the same point in output space (same category), then simple techniques, such as learning with a small, fully-connected backprop net, would do the job. Alas, accurate mappings for this task are much nastier: even characters that appear essentially the same to a human can be far apart in a simple metric where 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 can't provide acceptable results. This is because in such a general network, an enormous amount of training data is needed to determine the huge number of free parameters that set the network weights. With limited data, these 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. This is called feature extraction. In a series of character recognition experiments, we used a set of hand-crafted feature extractors to provide feature map representations. In an other solution, we designed a network architecture, also based on feature maps, but where the features were learned.

A Pre-Programmed Feature Extraction System

In an early attempt at hand-printed digit recognition, we used a series a heuristic methods, and hints from biological vision systems to develop feature extractors. The original grey-scale images were thresholded and size normalized to provide a 32 x 32 binary image. Next, we "skeletonized" the images, reducing the stroke width to one pixel. We then convolved the skeletonized images with 49 feature extraction kernels that were chosen by hand. These included families of oriented line and arc detectors, repeated at several scales. The resultant 49 maps were thresholded and combined by logical operations to produce 18 32 x 32 maps. To

reduce the amount of data, each feature map was coarse-blocked into a 3 x 5 array by simply OR-ing neighboring bits in the larger maps. (The coarse blocking also has the advantage of building a fair amount of translation, rotation, and distortion invariance into the processing.) The 18 3 x 5 maps then constituted a 270-bit vector that was used for classification. We used statistical techniques, such as Parzen windows, or a small backpropagation network with one hidden layer for the classification step. The backprop net had slightly higher accuracy than the other methods we tried, but its prime advantage was that it ran quickly on our workstation.

The above recognition system had state-of-the-art classification accuracy. The classifier was trained on feature maps extracted from images of 7300 zipcode digits taken from envelopes by a U. S. Postal Service contractor. It was then tested on 2000 different digits obtained in the same way. On the test set, the system achieved a substitution error rate of 1% while rejecting 11% of the digits as unclassifiable. Although this level of correct classification may, at first glance, not seem very remarkable, inspection of typical digits taken from the test set and shown in Figure 1, illustrate how hard the task really is. Many of the digits are poorly formed, have extraneous markings, or are in other ways so ambiguous that it is hard even for a human to classify them with confidence.

The major computing tasks in the above system were the skeletonization and the feature extraction. Both these tasks can be reduced to a series of convolutions of the image, where only a small part of the input data is processed at any given time. For this task, we used a custom chip to convolve simple kernels across the image at high speed [3] and threshold the output. In fact, the kernels were chosen to be compatible with the chip design which allowed kernels up to 7 x 7 in size with 3-valued coefficients. This feature extraction process was equivalent to the evaluation of a network with five layers of weights: three for skeletonization and two for the actual feature extraction and coarse-blocking. Although this equivalent network has about 4,000,000 connections, most of the processing fits on a chip with only 3,000 weights. This was possible because, at most, only a 7 x 7 region of the image was needed at any one time to determine the value of a unit in the next layer.

The time bottleneck for this system was the formatting of the data sent to the chip. The chip was designed as a general pattern-matching circuit without much thought of how it might be used in a large image-processing problem. In evaluating a convolution, the kernel is scanned serially across the image, with the center of the kernel passing over each pixel location. Thus, for a 7 x 7 kernel, the same pixel is presented 49 times, each time in a different location in a 7 x 7 array. In a chip designed as a convolver, shift registers

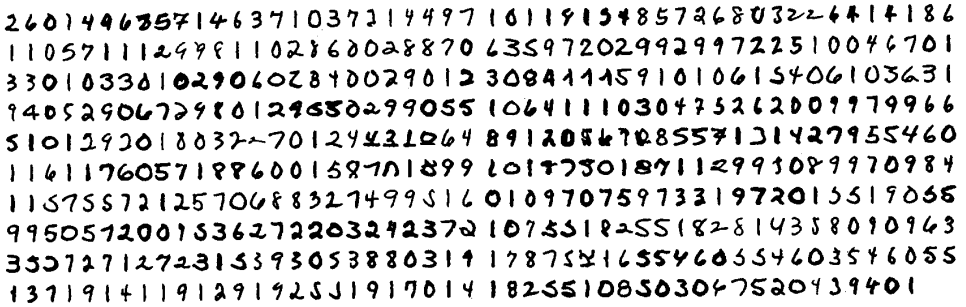

Figure 1. Typical examples from the U.S. Postal Service handwritten digit database. Notice that many of the digits are poorly formed and hard to classify, even for a human.

are provided, so that the same data need not be passed onto the chip more than once. Our general purpose chip had no such register, and to evaluate the convolutions we had to pass 49 times more input data than algorithmically required. Furthermore, the chip had no convenient way of storing intermediate results, so the data from the evaluation of each layer had to pass off the chip and on again before the next evaluation. Learning from this experience, our new chips include both these features.

Despite the inefficiencies in this system, it could classify about one character/sec (including preprocessing), evaluating over 8 million connections/sec. A workstation, doing all the same processing in software ran about 30 times slower. In fact, without the chip to help us devise the feature extraction process, limits to our patience would have prevented us from ever designing a high-performance recognizer.

A Digit Recognizer for Learned Feature Extraction

The success of the chip-based system with hand-crafted feature extraction and the poor performance of systems which attempted to do recognition using backpropagation learning directly on a pixel image might lead to the conclusion that there is no particular advantage to a neural-net approach, except as a language for describing a hardware accelerator. Being even more hard-nosed, the chip itself is really just used as an image convolver with a thresholded output.

Such a conclusion, it turns out, would have been premature. Recently, Le Cun [4], working with a small database of digits, showed that a network could be *trained* to develop its own feature extraction kernels. He designed a small network with a limited connectivity where the first few layers performed multiple convolutions. This was guaranteed by constraining the weights in the net to take the form of convolution kernels, but *not fixing* these weights. Le Cun's 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. These ideas were refined and applied to a larger network that was trained on the Postal Service data with astonishing results: the classification accuracy was about the same as the net with the hand-crafted feature extraction, but the new net required less than 70,000 connections (a factor of 30 less than the hand-crafted net) and had only 10,000 free parameters [5]. It is shown schematically in Figure 2. 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 the scaling of the images. The network was refined by deleting low-importance weights [6], and replacing many low-resolution feature maps by a few higher-resolution maps followed by adaptive coarse-blocking. This net, which had only 2500 free parameters, gave even better performance, rejecting 9% of the test digits to attain a 1% substitution error[7].

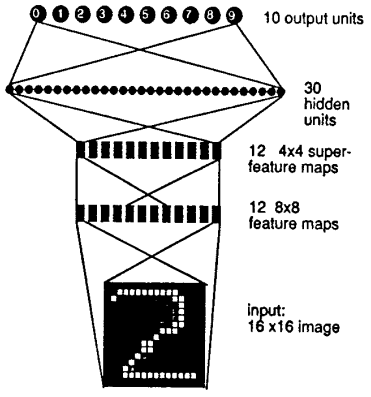

Figure 2. A network architecture for classifying handwritten digits. All the connection weights between layers are learned using the backpropagation algorithm. Two layers of feature extraction are followed by an addition layer of 30 neurons (here labeled hidden units). In later networks these 30 neurons were eliminated, and the first layer of 12 feature maps was replaced by a layer of 4 higher-resolution maps.

Common Characteristics of the Recognizer Networks

After looking at the example systems just described, common themes in the architecture emerge. While we can't make the leap to say that they *must* be common to all machine perception systems, it's a good bet that they will be important.

Here is what we find: the early layers of the network have limited receptive fields. 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 important, units within a map perform the same operation as their neighbors, only they operate 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. Since high-level features require less precision in their location, the networks also tend to have contractions of the feature maps, so that the maps have less spatial resolution than their inputs.

That biology seems to have come up with the same principles in animal vision systems, where numerous feature maps are formed for the visual field, gives us 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 fast, perhaps 100,000 times faster than biological neurons. Clearly, systems using electronic neurons would do well to compensate for their limited parallelism with time-multiplexing to take 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, unlearned 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.

An additional result, confirmed both empirically and theoretically, showed that as we move closer to the input of the network, the connections can have fewer and fewer bits of analog depth and still preserve the accuracy of the classifier. This means that at the front-end of the net, which has the greatest share of connections, we can use hardware with only a few bits of precision. Here, a high-density analog processor is appropriate. As we near the output layer, there are relatively few connections, but they need high precision. At this point, a good choice for hardware is a digital signal processor (DSP) which has high precision and is still an order of magnitude faster than a general purpose microprocessor.

An Advanced Neural-Net Chip For Machine Vision

In this section we describe a second generation neural-net chip [1], 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.

Figure 3 is a photograph of this chip which measures 4.5 mm by 7.0 mm. It 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, which appear as the rectangular blocks on either side of the chip, each take up a cell 23 microns by 17 microns and contain 9 transistors. The block in the center of the chip contains thresholding circuitry, multipliers, switches to set the network architecture, and decoders. At both ends of the chip are input and 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. Digital methods are used to 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. The chip has now been used to do edge extraction from images and is being incorporated into a machine vision system.

In our most advanced character recognizer, the DSP evaluates the network, but it spends most of its time scaling the image. Therefore, using the high-speed chip to evaluate the network would only result in a small speed improvement. However, with the chip's high speed, we may be able to avoid scaling the image; we can extract features across the entire input field at all scales. Furthermore, the image segmentation, which is also a computational bottleneck, need not be done explicitly. We can have many networks scanning portions of the image, attempting to recognize whatever appears in their field of view.

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.

Acknowledgement

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

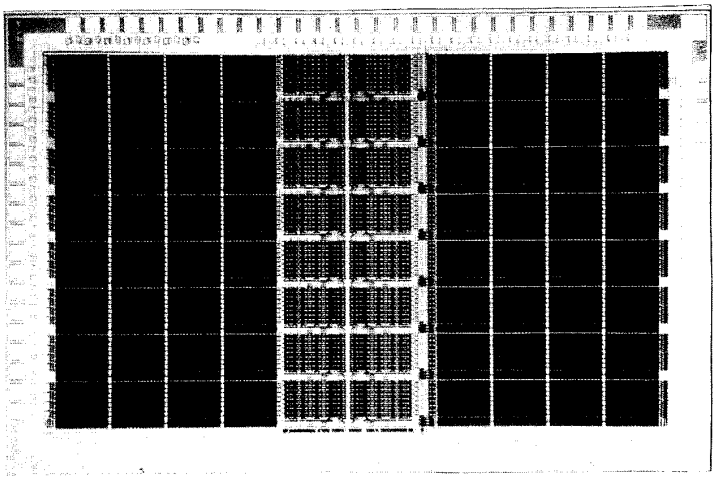

Figure 3. A photograph of a new 32,000 connection neural net chip. The chip size is 4.5 mm by 7 mm and can process 300 billion connections/sec.

References

[1] H. P. Graf and D. Henderson, "A reconfigurable CMOS neural network", to appear in Technical Digest of International Solid-State Circuits Conference, IEEE, 1990.

[2] S. Kahan, T. Pavlidis, and H. S. Baird, "On the Recognition of Printed Characters of Any Font or Size", IEEE trans. on PAMI, **PAMI-9** pp. 274-288 (1987).

[3] 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. 323-31 (1989).

[4] Y. Le Cun, "Generalization and Network Design Strategies" in *Connectionism in Perspective*, R Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, eds. Elsevier, Zurich (1989).

[5] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Back-Propagation Applied to Handwritten Zipcode Recognition" to appear in Neural Computation **4** January, 1990.

[6] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal Brain Damage", to appear in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, ed. Morgan Kaufman, San Mateo, CA, 1990.

[7] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a backpropagation network" to appear in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, ed. Morgan Kaufman, San Mateo, CA, 1990.