OPTICAL CHARACTER RECOGNITION: A TECHNOLOGY DRIVER FOR NEURAL NETWORKS

R. E. Howard, B. Boser, J. S. Denker, H. P. Graf
D. Henderson, W. Hubbard, L. D. Jackel, and Y. Le Cun
AT&T Bell Laboratories, Room 4E-436
Holmdel, NJ 07733
and
H. S. Baird
AT&T Bell Laboratories
Murray Hill, NJ 07974

Introduction

Over the last several years, computing systems based on adaptive learning with fine-grained parallel architectures have moved from obscurity to front-page prominence. These systems derive some of their novel architecture from ideas gleaned from biology, hence the name "neural network". Although many of the ideas behind this field are not new, improved computing hardware, better understanding of learning algorithms, and limitations of traditional approaches have combined to renew interest in neural nets.

The ultimate success of electronic neural networks will depend on their effectiveness in solving real-world problems. Therefore it is important to choose realistic benchmarks as a focus for research in algorithms and hardware for neural-network computing. Optical character recognition (OCR) of handwritten digits is such a benchmark problem: it has a clearly defined commercial importance and a level of difficulty that makes it challenging, yet it is not so large as to be completely intractable.

We have demonstrated that a neural net can perform handwritten digit recognition with state-of-the-art accuracy. The solution required "automatic learning" and generalization from thousands of training examples, and also required designing into the system considerable knowledge about the task --- neither engineering alone nor learning from examples alone would have sufficed. The resulting network is well-suited for implementation on workstations or PCs, and can take advantage of digital signal processors (DSPs) or custom VLSI.

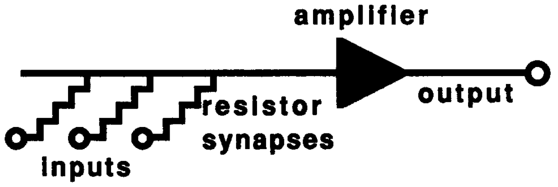

Figure 1. An electronic neuron model. The amplifier receives signals to its input through resistors connected to the outputs of other amplifiers or external inputs. Similarly, the amplifier output is connected to other amplifiers through their input resistors.

Electronic neural models and networks

Figure 1 shows an electronic model of a neuron --- a greatly simplified abstraction having some of the basic functions observed in biological systems. Input signals are represented as voltages and the resulting currents through the resistive "synapses" are summed on the input wire to the nonlinear amplifier. Negative (inhibitory) connections can be obtained using a differential amplifier or a current mirror. The input voltages can be considered the components of a vector (the input vector), and the synapse conductances can be considered the elements of a vector (the weight vector); the neuron begins by calculating the dot product of these two vectors. The product is applied to the input of the amplifier. If it is either very large or very small, the amplifier saturates; for smaller inputs the transfer function is approximately linear.

Several such neurons, sharing the same inputs, can be grouped together to form a "layer" of neurons. An electronic neural network consists of one or more of these layers feeding forward (and perhaps backward). With appropriate synapse weights, it is easy to construct a neuron that functions as a NOR gate, which implies that a two-layered network can implement any Boolean function. In fact, layered networks are much more powerful; a three-layer network can implement essentially any reasonable analog or digital input-output relation.

A single layer network can classify objects when the boundaries between the classes can be represented as linear functions in the space of the input examples. Multilayer networks can make far more complex partitions of the input space. Such networks offer the promise of being useful for difficult classification tasks like speech and vision processing [1], where very dissimilar input patterns must often be put in the same output category.

Neural Network Hardware

The electronic neural model of Figure 1 offers several arguments in favor of analog computation. Unfortunately, the situation is not as simple as it first appears for several reasons. First, any network large enough for a real-world problem contains enough neurons and synapses (10,000+) to challenge existing analog VLSI technology. Second, in analog circuits only limited precision is available, and it is very difficult to transfer analog signals from chip to chip in a large system. High precision could require a large, slow circuit that cannot compete with straightforward digital designs.

There are several other factors that make digital implementations of neural networks competitive. Modern workstations provide reasonably good network development tools, and can now perform several million multiply-add operations per second. (This measure of performance is usually called a "connection per second" in the language of neural computing.)

Higher processing speeds are possible using digital signal processors (DSPs), which are optimized to perform multiply-add operations. They can achieve over 20 million 32-bit floating point connections per second, and provide 10 times the processing power at 1/10 the cost of a typical workstation. Singly or in arrays, they offer the possibility of cost-effective implementation of neural nets.

Only when the utmost performance is required do analog solutions stand out. Our group and others have designed chips incorporating analog processing for the multiply/add operation and digital processing for I/O and control functions. Speeds of several hundred billion connections per second at low resolution (1-4 bit) have been achieved [2]; typical applications can only utilize a fraction of this power because of bottlenecks in the preprocessing of the data.

Adaptive systems and Learning

A key advantage of neural-net methods is the possibility of "adaptation" and "learning from examples". Neural-net learning algorithms that run on general purpose computers can determine good choices for network architectures and weights. The resultant network can then be installed on dedicated neural hardware or be run on a general-purpose machine.

Analysis shows that many difficult pattern-recognition problems can be formulated as multi-dimensional curve fitting. For example, in OCR, general rules for distinguishing one character from another are not known. The best we can do is to examine many characters written by a cross-section of the population and try to find a function that interpolates (generalizes) adequately to allow recognition of new examples of the same general type. Unfortunately, any function that maps character images to one of ten output categories must be extremely complex and have a tremendously high-dimensional input. Most curve fitting and statistical-inference procedures work best in low dimensions, and are next to hopeless for our application. However, there exist neural-net learning algorithms that perform this interpolation surprisingly well.

The network cannot learn effectively from examples unless it starts with some knowledge of the problem to be solved. Therefore we require a method for programming this knowledge into the network. This is in stark contrast with conventional programming languages or rule-based systems, where engineering knowledge can be incorporated easily, but where improving the system by adaptation is very difficult.

Most neural-net learning today is performed using the algorithm known as "Back-Propagation". The algorithm requires thousands of training examples, each labeled with the corresponding desired output. Each training example is presented to the network, and small changes are made in the synaptic weights, to force the network's output closer to the desired answer. The process is repeated until a set of weights is developed that allows the network to classify accurately all the examples in the training set.

The resulting solution is not unique, and there is often no guarantee that it will generalize to a wider set of examples. If the network is too small or too large or has the wrong topology, or if the training set is inadequate, the network solution will generalize poorly. However, given a network architecture and constraints that well reflect the problem at hand, and given a large, rich training set, it is possible for the network to achieve truly remarkable performance. In some cases solutions are possible that surpass conventional techniques in accuracy and require far less development time.

Design of an Optical Character Recognition System

In this section we describe neural-net approaches to handwritten digit recognition and show how advances in theoretical understanding of neural-net learning has led to improved system performance.

The key to tackling any large problem with an adaptive system is to obtain a large and representative database. For our application, we used a collection of zipcode digits collected from real US Mail envelopes. A US Postal Service contractor converted the original images to a digital format and segmented them into isolated digits. The resulting database consisted of 9,298 binary images of isolated digits, 7291 of which were used for training; the other 2007 were reserved for testing. Before presenting the images to the network, additional preprocessing was done to ensure uniform scaling and orientation, and to correct for noise and broken lines. The resulting processed images are presented as vectors to a network that is either implemented in software or a mix of custom hardware and software. Figure 2 shows examples of the normalized digits used.

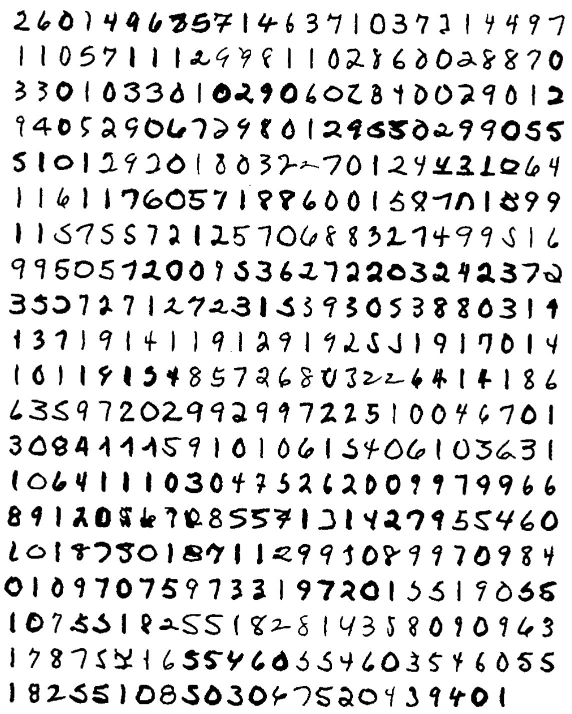
Figure 2. Typical examples from the U.S. Postal Service handwritten digit database. Notice that many of the digits are poorly formed and hard to classify, even for a human.

Just feeding an input image into a multilayer, fully-connected network gives a poor solution to this problem. If the network has a sufficient number of units in the intermediate layers to be able to fit the training data, it will have so many adjustable parameters (synapse weights) that the generalization will be underdetermined. The analogous problem arises in ordinary 2-dimensional curve fitting: if too few data points are fit to a high-order function, the curve may well pass through every data point but have disastrously poor interpolation properties.

Better generalization results are obtained using a "constrained" network that incorporates our knowledge about the problem into its structure. Since we are processing 2D images, it is important to have a network structured to organize the input in a 2D space. We have therefore designed networks that perform multiple 2D convolutions of the image to extract geometric features and form a family of 2D feature-maps. The coefficients that define the kernels of these convolutions are the adjustable parameters of our network. We find that these kernels typically serve as feature detectors for simple, perceptually important features such as oriented lines and edges. A map is created for each class of features. For our application, these maps are a better representation of the data than the original pixel map. In particular, distortions in shape or location of a digit usually have a smaller effect on the feature maps than on the original pixel image.

Our first successful attack at this problem involved using a custom chip suitable for evaluating simple 3-valued kernels convolved across an image at high speed [3] and thresholding the output. In this case, the features were chosen by hand, using some weak hints from biological vision systems and simple heuristics. The chip was also used to "skeletonize" the images, that is to make the lines one-dimensional, removing meaningless linewidth variations. After these two levels of preprocessing, the resulting feature maps were presented as input to an additional 3 layers of interconnected neurons trained using the backpropagation algorithm [4]. The total network is equivalent to almost 3,000,000 connections and 7 layers (since the skeletonizing layers were repeated several times). More than 99% of these connections were tailored to fit on the custom chip as 3-valued weights with binary outputs. Because of the large number of repetitive kernels, the network had only about 13,000 free parameters, which is less than 1% of the number of connections that must be evaluated. The performance was measured by choosing a confidence threshold on the output layer to reject a sufficient number of digits so that the error rate on those remaining was 1%. For this network, that reject fraction was about 11% of the 2007 digits in the test set.

In the above network, the 49 features used were chosen by hand, without automatic adaptive learning. A better network was designed by replacing these binary features with a smaller number of features with greater dynamic range and using back-propagation learning to adjust the kernel weights. The required dynamic range exceeded the capabilities of our currently available chips, so the net was first implemented on a general purpose computer. (Chips adequate for this net are now being tested and should provide orders-of-magnitude increases in speed.) Figure 3 shows a schematic of one such network with 66,000 connections and about 10,000 free parameters. This network has 2 layers, each with 12 feature-extraction kernels that are learned during the training process. Although this network is smaller than the previous design, it makes more efficient use of its resources and its reject performance is about 11%. The smaller size means that the computation time on a general purpose processor is reduced by almost a factor of 40 compared with the earlier network. By replacing the original custom hardware by a digital signal processor chip, this network can classify more than 10 digits per second (including all preprocessing, which is performed by a personal computer) [5]. With a second generation custom chip optimized for these higher resolution kernels and with pipelined I/O, another speedup by a factor of 10 to 100 is expected [2].

Further improvements in performance in the system have been obtained by selectively removing connections. We have developed an automatic process to prune weights from an existing network based on their importance [6]. During training, the back-propagation algorithm adjusts each weight so that any change in the weight has no first-order effect on the output. The system is at a local minimum in "weight space". One can then measure the second derivative of the output with respect to each weight and determine which ones can be removed without serious effect. Using this algorithm, we were able to delete the 30 hidden units of Figure 3, and decrease the number of first-layer feature maps from 12 to 4. These 4 maps have higher resolution than those in the previous design, bringing the total number of connections to 98,000. However, the new network has only 2700 free parameters, a factor of four reduction from the parent design. As the number of free parameters decrease performance improved, reaching a reject rate of about 9%. At this level, this network is a state-of-the-art classifier of handwritten digits [7]. Further reduction in the size from this point led to degradation in performance.

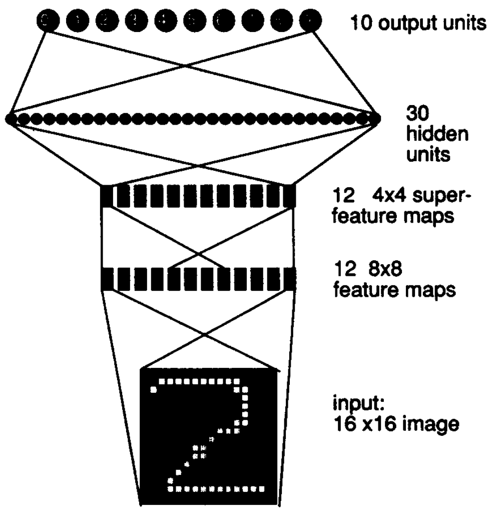

Figure 3. A network architecture for classifying handwritten digits. All the connection weights between layers are learned using the backpropagation algorithm. Two layers of feature extraction are followed by an addition layer of 30 neurons (called hidden units). In later networks these 30 neurons were eliminated, and the first layer of 12 feature maps was replaced by a layer of 4 higher-resolution maps. These changes were motivated by a weight pruning process described in the text.

This trend of improved performance as the network size is reduced is easily understood through the curve-fitting analogy. Making the network smaller is equivalent to reducing the order of a curve fit. If it is done using the proper family of functions, the interpolation properties of the fit will improve as the number of parameters is reduced. Eventually, the fit will reach an optimum (model and data dependent) below which there are insufficient degrees of freedom to fit the data. For neural networks, the topological and other constraints placed on the network determine what family of functions will be implemented. A rough rule of thumb is that the number of adjustable parameters should not exceed the amount of available training data.

Conclusion

Neural-net methods have now demonstrated their usefulness for tackling hard, practical problems. Neural-net systems now exist for handwritten digit recognition that have state-of-the-art performance. Significant improvements in recognition speed will occur when new custom neural-net chips are incorporated into the systems. Research in real-world problems like optical character recognition identify key issues for neural-net hardware and provide a proving ground for algorithm development in ways that can't be achieved with toy problems.

Acknowledgement

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

References

[1] For a general introduction see:"Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations," Rumelhart and McClelland (Eds.), Bradford Books/MIT Press (1986).

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

[3] H. P. Graf and P. deVegvar, "A CMOS Associative Memory Chip Based on Neural Networks," in Technical Digest International Solid-State Circuits Conference IEEE, (New York, 1987), p. 304.

[4] 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).

[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", Neural Computation, 1(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.