LEARNING INTERNAL REPRESENTATIONS FROM GRAY-SCALE IMAGES: AN EXAMPLE OF EXTENSIONAL PROGRAMMING

Garrison W. Cottrell
Institute for Cogntive Science
University of California, San Diego

Paul Munro
Department of Information Science
University of Pittsburgh

David Zipser
Institute for Cogntive Science
University of California, San Diego

ABSTRACT

The recent development of powerful learning algorithms for parallel distributed networks has made it possible to program computation in a new way. These new techniques allow us to program massively parallel networks by example rather than by algorithm. This kind of *extensional programming* is especially useful when there are no known techniques for solving a problem. This is often the case with the computations associated with basic cognitive processes such as vision and audition. In this paper we apply the technique to the problem of learning an efficient internal representation of image information directly from a gray-scale image. We compare the results of this to the engineering version of this problem, i.e., image compression. Our results demonstrate that a very simple learning method learns internal representations that are nearly as efficient as those developed by the best known techniques in image compression. Thus we have a technique whereby neuron-like networks can self-organize to form a compact representation of a visual environment.

LEARNING INTERNAL REPRESENTATIONS FROM GRAY-SCALE IMAGES: AN EXAMPLE OF EXTENSIONAL PROGRAMMING

Garrison W. Cottrell
Institute for Cogntive Science
University of California, San Diego

Paul Munro
Department of Information Science
University of Pittsburgh

David Zipser
Institute for Cogntive Science
University of California, San Diego

INTRODUCTION

The recent development of powerful learning algorithms for parallel distributed networks has made it possible to program computation in a new way. These new techniques allow us to program massively parallel networks by example rather than by algorithm. This kind of *extensional programming* is especially useful when there are no known techniques for solving a problem. This is often the case with the computations associated with basic cognitive processes such as vision and audition. In this paper we apply the technique to the problem of learning an efficient internal representation of image information directly from a gray-scale image. We compare the results of this to the engineering version of this problem, i.e., image compression. Our results demonstrate that a very simple learning method learns internal representations that are nearly as efficient as those developed by the best known techniques in image compression. Thus we have a technique whereby neuron-like networks can self-organize to form a compact representation of a visual environment.

The technique we employ is known as *back propagation*, developed by Rumelhart, Hinton, and Williams (1986). While we will not go into the details of it here, back propagation can be considered a generalization of the perceptron learning procedure for multilayer nonlinear networks of neuron-like computing elements. Training the network consists of repeated presentations of input-output pairs representing the function to be learned. The learning algorithm operates by adjusting the weights between the elements of the network in such a way as to reduce the overall error in the output. In many cases, the network finds a solution to the problem that was unknown in advance to the user. In doing so, it develops its own *internal representation* of the input that is useful for solving the problem. It is often difficult to analyze this representation because many units are involved and the representations are highly *distributed* over the set of internal units. A subgoal of the present research is to make a first step towards unraveling the nature of these representations by applying the learning mechanism to a domain where the types of useful representations have been well studied.

Another aspect of this work is that the representation of images in an efficient format by neuron-like computing elements may give us clues to the way such information is represented in actual neural tissue. The learning procedure itself is not particularly biologically plausible, but the mechanisms it discovers for solving problems are (Zipser, in press). Whether or not there is anything like back propagation in the brain, we learn something about how the brain *could* solve problems from the "neural" solutions it discovers. Such information could be useful in guiding neurobiologists in their observations of cell firings during cognitive tasks.

Encoder Networks

The problem of finding an efficient internal representation of an environment is called the *encoder problem.*[1] In PDP networks using back propagation, this problem is solved by giving a network the problem of performing an identity mapping over some set of inputs. The network is constrained to perform this mapping through a narrow channel of the network, forcing it to develop an efficient encoding in that channel. There are two interesting aspects to this: (a) the network is developing a compact representation of its "environment"; and (b) although the algorithm used was developed as a supervised learning scheme, in this case the learning can be regarded as unsupervised—since the training signal is the same as the input, the system self-organizes to encode the environment.

A network appropriate for performing this task in the image domain is shown in Figure 1. It consists of an 8×8 input patch, corresponding to a two-dimensional patch of an image, that is completely connected to sixteen *hidden units*, the "narrow channel" through which the patch of image must be transmitted. These hidden units are completely connected to an 8×8 output patch, where the image is reconstructed.

PROCEDURE

We trained the above network with a digitized image of the Intelligent Systems Group (ISG) at UCSD (Figure 2). A digitized image is an $M \times N$ light intensity function $f(x, y)$, where x and y correspond to the spatial coordinates within the image, and $f(x, y)$ is a light intensity value from 0 to 255. One element of $f(x, y)$ is often referred to as a *pixel*, for *picture element*. Thus the original image has eight bits of information for each pixel. However, there is a great deal of redundancy in this information. Neighboring pixel values will tend to be highly correlated. If the network can capture this redundancy, it can represent the image more compactly.

We trained our network by randomly sampling 8×8 patches of this image, converting the gray level value linearly to the range [0,1].[2] These values form the input to the network. Activation passes through the net, and

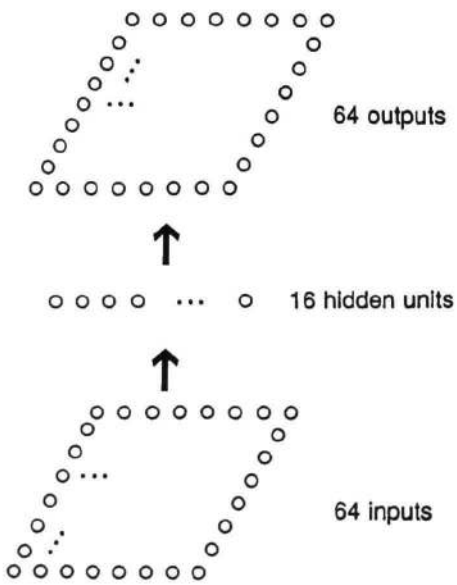

Figure 1. The network used in most of our examples.

[1] Ackley, Hinton, and Sejnowski (1985) were the first to demonstrate a learning algorithm for PDP networks that could solve the encoder problem.

[2] We used the usual sigmoidal activation function, with the output range scaled to [-1,1]. Since this function only asymptotically achieves the end values, it is easier for a unit to achieve values in the middle of the range. Hence converting the gray scale values to the range [0,.85] works better on this problem. We show results from the [0,1] conversion for historical reasons.

Figure 2. The original image of the Intelligent Systems Group (ISG) at UCSD.

activation of the output patch is obtained. This is compared with the input value, error is propagated back through the network, and the weights are updated according to the back propagation algorithm. We used an initial learning rate of .25 (no momentum), and trained the network on 100,000 patches of the image. Then the learning rate was lowered to .01 and the network was trained for an additional 50,000 iterations.

The result of this training is a "patch compressor." A reproduction of the image can be obtained by systematically applying this patch compressor across the original image, reconstructing a (nonoverlapping) patch at a time. In this way, the entire image is passed through the narrow channel of the hidden units, and we can view the reconstructed image to get an idea of the fidelity of the representation obtained by the hidden units.

In order to compare our results to that of image compression techniques, it is necessary to obtain a comparable measure of the number of bits used to represent the image. Image compression is measured by the number of bits transmitted per pixel of the reproduced image. In our case this corresponds to:

$$bits/pixel = \frac{(bits/hidden\ unit\ output)\times(\#\ hidden\ units)}{\#\ of\ pixels\ reproduced}$$

We must *quantize* (round off to a fixed number of values) the outputs of the hidden units in order to use this formula. For example, if we round off to 32 different output values, then this corresponds to five bits per hidden unit output. In the following examples, we used a uniform quantizer—the rounded-off values are equally spaced between [-1,1]. We could have done better (in terms of resulting error) by quantizing in ranges where the hidden unit outputs spend most of their time.

Finally, we need an objective *fidelity criterion* to measure how close the reconstructed image is to the original. The standard measure used is the mean square error normalized with respect to the squared intensity of the image. If $g(x, y)$ is the reproduced image, then the error is given by

$$e(x, y) = g(x, y) - f(x, y) .$$

the mean-square error is given by

$$MSE = \frac{1}{M'N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} e^2(x, y) ,$$

and the normalized MSE with respect to the average squared intensity of the image is given by

$$NMSE = \frac{MSE}{\frac{1}{M'N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f^2(x, y)} .$$

This is what we will use, expressed in percent.

RESULTS

The problem is to develop an efficient representation of the information in a digitized image. The network of Figure 1 does this by processing repeated presentations of samples of the visual environment (the image in Figure 2), using back propagation to correct the internal representation. The result is that the image can be represented with very little loss of information with 1 bit/pixel, representing an eight-fold compression of the information in the image. Also, the same representation does a good job of reproducing several images the network was not trained on.

Some reconstructions of the ISG image are shown in Figure 3. In Figure 3A five bits of hidden unit output were used, representing 1.25 bits/pixel. The most noticeable degradation from the original image is that the stripes on the shirt of the seated gentleman (Don Norman) are gone. This is not noticeably different from the result if we do not quantize the hidden units. On the other hand, reducing the output levels by another bit (16 values) more noticeably degrades the image (Figure 3B).

It turns out we can recover the shirt stripes if we use more hidden units, but compression suffers. Figure 4A is a reconstruction using 32 hidden units, with 16 output values each. This represents a compression of 2 bits/pixel. Higher compression can be obtained by using fewer hidden units, but the result is less satisfying. Figure 4B shows the results of using a network with 8 hidden units and 32 output values, resulting in .625 bits/pixel. More examples exploring the space of numbers of hidden units vs. numbers of quantization levels can be found in (Cottrell, Munro, & Zipser, in press).

How good a representation is this for images other than the training image? We naively expected that perhaps a network could be trained that would work well for all images, justifying the expense of the initial training. This is a somewhat misplaced dream, given that our network learns, in some sense, the statistics of the image it is trained on, and different images have different statistics. However, it may work well for a class of images. It turns out that it does a good job of reproducing some images that it wasn't trained on. Two of the images we tested it on and their reproductions are shown in Figure 5. We expect that it would not work well for images with very different statistics, such as text, but have not had a chance to try it on such images yet.

The Internal Representation

What is the internal representation at the hidden unit layer? Figure 6 shows the internal representation for eight hidden units. Each row corresponds to one hidden unit. Figure 6A shows the weight matrix for each of eight hidden units thresholded at various levels, one hidden unit per row. The center column, representing a threshold of 0, identifies which weights are negative and which positive. This gives an idea of the kind of pattern that excites each hidden unit the most. Figure 6B shows the output patch driven by each hidden unit alone. Again, each row corresponds to one hidden unit, and the columns correspond to different levels of activation from the hidden unit. The right-hand column thus corresponds to the output weights from that hidden unit. One obvious thing to note here is that the hidden units try to reproduce what they "see." Figure 6C shows the same information as 6B, in a gray scale image (6B is a thresholded version of 6C).

Figure 3. Quantization effects. A: 5 bits, 1.25 bits/pixel, NMSE 0.474%. B: 4 bits, 1 bit/pixel, NMSE 0.676%.

Figure 4. A: The reproduced image using 32 hidden units, four bits of quantized levels, 2 bits/pixel, NMSE 0.625%. B: The reproduced image using eight hidden units, 32 quantizer levels, resulting in .625 bits/pixel, NMSE 1.182%.

A

B

Figure 5. Two images (on this page and next) reproduced by the network trained on the image in Figure 2. A: The Symbolics Graphics group. B: Reproduced image, using six bits of quantized values, 1.5 bits/pixel, NMSE 1.267%.

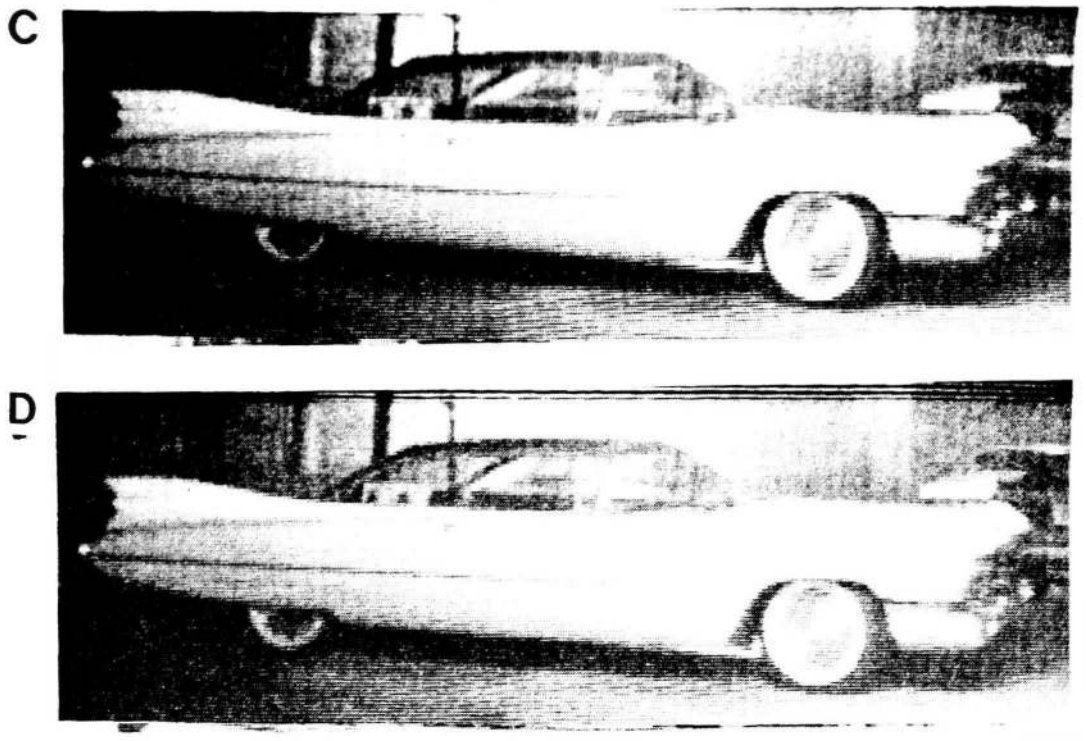

Figure 5. C: Cadillac. D: Reproduced Cadillac, 1.5 bits/pixel, NMSE 0.764%.

What do these weights represent? We don't have an analytic answer to this question. However, we can compare the network's solution to a standard technique, the Principal Components Transform (PCT), to get an idea of what it does.

First we set up some correspondences between our network and the usual image compression system. The first step in a transform encoding system is to multiply the patch vector by a matrix to obtain less correlated coefficients:

$$\mathbf{y} = \mathbf{A}\mathbf{x}.$$

The y_i's are sent through a channel in a coded form, and at the other end they are transformed back into image space. The reconstructed image is the inverse transform

$$\hat{\mathbf{x}} = \mathbf{A}^{-1}\mathbf{y}\ .$$

It is the form of **A** that determines the type of transform. In the principal components transform, the rows of **A** are the eigenvectors of the covariance matrix of the **x** patch vector. This corresponds to setting up a new coordinate system with axes along the directions of maximum variance, and sending the coordinates in this new system. Then the inverse matrix converts back into image coordinates. For a principal components transform, this inverse matrix is just the transpose of **A**. What is often done in this case is to just send the coordinates along the first k dimensions—the ones with highest variance. What this means is that the coefficients themselves (the coordinates along these high-variance axes) also have variance that is high for the first coordinate and that monotonically decreases.

The analog in our network is that **A** is the weight matrix between the input and hidden unit layers, with each row of **A** corresponding to the input weights on one hidden unit, and each hidden unit output a semilinear version of y_i. Similarly, the weight matrix between the hidden units and the output patch corresponds to $\mathbf{A}^{-1}$.

Now, we can begin to understand what the network does. First, observation has shown that during

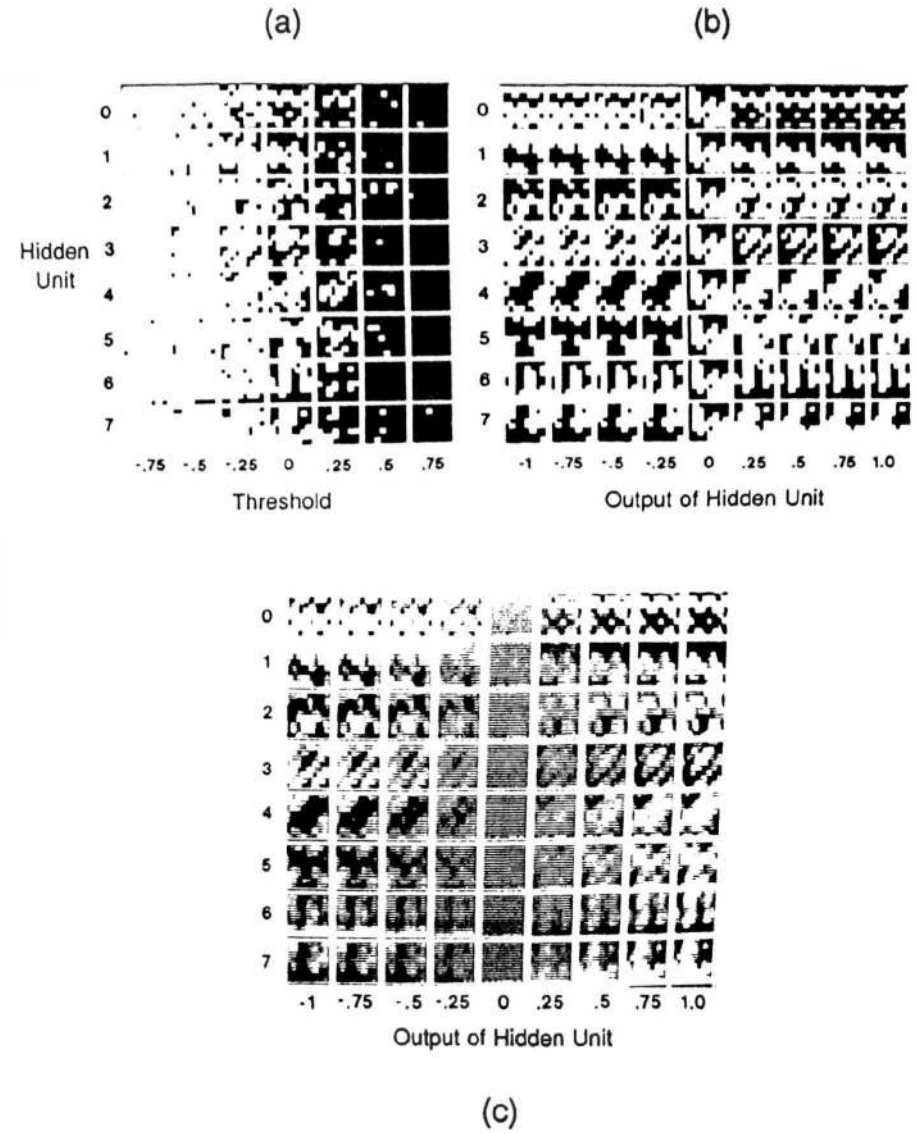

Figure 6. The internal representation. A: The weight matrices from the input patch to eight hidden units, thresholded from −.75 to +.75. The middle column (zero threshold) shows the "canonical" feature responded to by that hidden unit. B: The output patch driven by each hidden unit at different output values from −1 to 1. C: The same picture as (B) on a color monitor.

reconstruction of an image, the hidden unit outputs are mostly in the linear range of the activation function. So the network makes little use of the nonlinearity. Second, note that Figure 6 shows that the network also uses the transpose of the input weights as the output weights.

Finally, notice that the final image can be regarded as a linear combination of *basis images*: one for each coefficient (or hidden unit output). For comparison purposes, Figure 7 shows the basis images from the principal components transform for a picture of a cameraman (from Gonzales & Wintz, 1977). Figure 6B, the last column, shows the same thing for eight units of our network. Unlike the principal components transform, there is no obvious way to order the basis images.

This is reflected in the variances of the hidden units' outputs: They are all about equal (to 0.1) and the amount of error in the output accounted for by each one is of comparable size. Back propagation has spread the error relatively evenly across the hidden units. In the principal components transform, the "hidden units" would

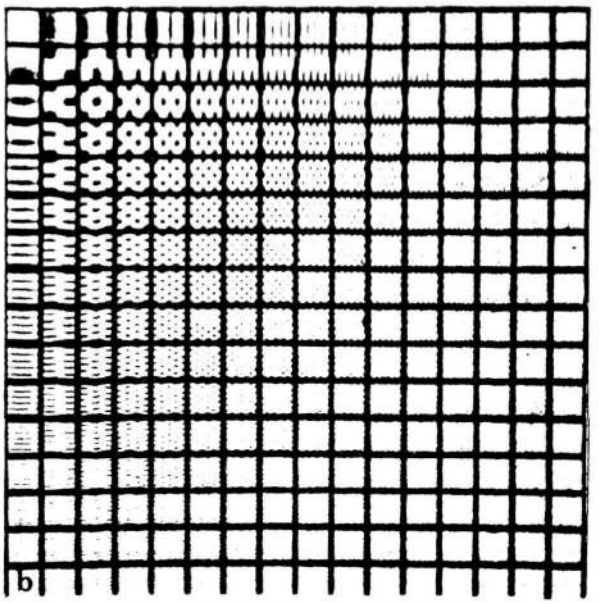

Figure 7. The set of Hotelling basis images for a particular image. (From Gonzales & Wintz, 1977. Reprinted by permission.)

have monotonically decreasing variance, and the variance typically falls off very quickly, so that they differ by orders of magnitude. Our conjecture at this point is that the hidden units span the space of the first several principal components, but are rotated so that each can have about equal variance.

DISCUSSION

This study has produced results that have implications for both connectionist networks and image compression. T' ese are discussed and summarized below.

Implications for Connectionist Networks

Extensional Programming

The major result of this study is that a relatively straightforward application of the back-propagation learning procedure to a problem that has been studied for many years results in near state-of-the-art performance. The key point is that this performance was obtained not by programming a connectionist solution to the problem, but by the process of *extensional programming*. In this procedure, many examples of the desired behavior are presented and the network must program itself to achieve the behavior. This suggests that other problems, where solutions are not known in advance, may be solved by back propagation.

A major problem with this technique is determining post hoc how the network solved the problem. In our case, we have some pieces of the answer, mainly because image compression is a well-studied problem. Hence we have some idea what to look for, if not an analytical solution. By comparison of our network to the techniques of image compression, we can gain insight into the solution found. However, this will not be the case in general. The importance of back propagation is that whether we know how to solve the problem or not, whether we know of an algorithm for the solution or not, back propagation will in most cases find a solution to the mapping simply from examples of the input-output patterns.

Another key point is that the network self-organizes to represent its environment. This is discussed elsewhere with relation to answering the question of how meaning might be grounded in perception (Chauvin, 1986; Cottrell, 1987).

Linear Networks

There is currently a bias in the connectionist community, shared by the authors of this article, against linear networks. This is partly due to the assumption that "interesting" problems must require nonlinearity for their solution. While the results were not reported here, we found that a linear version of the network produced results compatible with the nonlinear version. Since identity mapping is a linear problem this is not too surprising. However, it is useful to check whether nonlinearity is necessary for a particular problem. If not, the elimination of evaluating the logistic function can lead to more efficient solutions. If both approaches appear viable, comparison of the two can lead to a better understanding of nonlinear solutions, since the linear network lends itself to analysis much more readily than the nonlinear one (Williams, 1985). This approach needs to be carried further in future work.

Internal Representations

One of the typical ways to speak of the solutions discovered by back propagation learning is to say that the network discovers regularities in the input. This paper adds at least a new vocabulary for discussing the kinds of regularities discovered in the case of autocoding. We can look at the *variance* of the hidden units as indicative of their usefulness in the resulting solution. It may not be the case that hidden units span the principal subspace of the covariance matrix, that is, the space spanned by a PCT solution, but it is possible that the hidden units are finding the best approximation to this within the constraint that the logistic function imposes of a limited range on the coefficients. If this turns out to be true, then we may speak of the hidden units as finding at least an analog of the principal subspace and as discovering useful *axes of covariance* of the input.

Implications for Image Compression

A major result of this work is the application of a new way of minizing mean square error to a real-world problem that shows it is competitive with PCT. This new technique has several possible advantages over PCT and other current techniques. These need confirmation by further investigation.

One advantage is the relatively equal distribution of error among all of the coefficients. This should lead to a reduction in the effects of channel errors. In PCT and other techniques that approximate it, channel errors that affect the coefficients with high variance can result in a patch that is dominated by the corresponding basis image. The relatively equal contributions of each basis image in the network solution should mitigate these effects. In particular, we know in advance what range the value should be in, and if a coefficient is suspected of being in error, an acceptable restoration of the patch can probably be effected by simply eliminating that coefficient or replacing it with its average value.

Second, because of the fixed range of the coefficients, problems with "tracking" the coefficients by an adaptive quantizer is mitigated. Adaptive quantizers try to follow coefficients as they change, changing the quantization as the coefficients shift. They can "lose track." In our system, we know in advance the range of the coefficients, which should make this less of a problem.

Third, the ability of our network to generalize to novel images is striking. The performance of the linear network is especially encouraging in this regard. This requires some qualification. First, it is likely that this generalization does not apply to images with very different statistics, such as text. Second, we are not aware of work in this area investigating the ability of PCT to generalize to images other than the "training" image. Further work should compare these techniques.

CONCLUSIONS

The major result of this work is in demonstrating the efficacy of current connnectionist techniques for programming by example, rather than algorithm. We have termed this *extensional programming*. The results here suggest that this technique is a powerful one. Its naive application to a problem of current interest among engineers resulted in respectable performance compared to current methods.

However, back propagation is not a panacea—it brings new problems of its own. Designing a connectionist representation of the input (and output for nonautocoding problems) is itself an art. The representation must contain enough information to license solution of the problem, without providing so much that the solution is trivial. However, Hinton (1986) has shown that at least in some domains, back propagation can even design the input representation simply from the occurrence of a token in context.

The results of this study suggest that one useful approach to problems for which no algorithm is known, or for which no parallel algorithm is known, is to use connectionist representations of the problem and allow the network to discover the program itself. Analysis of the the programs thus discovered may aid in our understanding of the problem and lead to methods for doing the programming ourselves. A variety of problems that cognitive science is concerned with are of this character—the input-output behavior is known, but the algorithm is not. With extensional programming we can begin to investigate algorithms that we did not invent ourselves.

REFERENCES

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science, 9*,147-169.

Chauvin, Y. (1986). Hypermnesia, back propagation, categorization, and semantics. Preprints of the Connectionist Models Summer School, Carnegie-Mellon University, Pittsburgh, PA, June 21-29, 1986.

Cottrell, G. (1987). Toward connectionist semantics. In Y. Wilks (Ed.), *Theoretical issues in natural language processing: Position papers* (conference proceedings from TINLAP-3). Association for Computational Linguistics.

Cottrell, G., Munro, P., & Zipser, D. (in press). Image compression by back propagation: An exmaple of extensional programming. In N. E. Sharkey (Ed.), *Advances in cognitive science* (Vol. 3). Norwood, NJ: Ablex.

Gonzales, R. C., & Wintz, P. (1977). *Digital image processing*. Reading, MA: Addison-Wesley.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, August 15-17, 1986, Amherst MA.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1. Foundations.* Cambridge, MA: MIT Press/Bradford Books.

Williams, R. J. (1985). *Feature discovery through error correction learning* (Tech. Rep. 8501). La Jolla: University of California, San Diego, Institute for Cognitive Science.

Zipser, D. (in press). Programming neural nets to do spatial computations. In N. E. Sharkey (Ed.), *Advances in cognitive science* (Vol. 2). Norwood, NJ: Ablex.