Principal Component Analysis for Place Recognition

Jonathan Wang, Zachary Dodds, Willard Miranker[1]
Computer Science Department, Yale University
New Haven, CT 06512

Abstract
We present a hybrid neural network model to solve a place recognition problem. The front end is a self-organizing net equivalent to a principal component analyzer; the back end is a feed-forward net with backpropagation, i.e. supervised learning. A confidence level greater than 0.9 was reported as the net correctly recognized a repertoire of pictures it had not seen before.

1. INTRODUCTION

At the Yale Vision and Robotics Lab there is a Nomad robot that roams around the building taking pictures. It is desirable that the Nomad can recognize where it is by comparing a new scene with previously taken pictures. Formally, suppose we have images of M distinctive scenes. (We will use the terms "picture" and "scene" as synonyms for "image" throughout.) We seek an algorithm that will take a new image as input and determine which one of the M scenes the new image most resembles.

Here, we propose a neural network solution which combines stages of unsupervised learning and supervised learning. The network is composed of two independent subnetworks. The first subnet, which we call the "Principal Component Analyzer" (PCA) is self-organizing. It receives an image (the "input image,"possibly preprocessed) and outputs a set of real numbers. The latter express the most "important components" in the image. These "components" are the longest axes of the ellipse which bounds all of the images considered as data points in an appropriate, high-dimensional space. The second subnet, which undergoes supervised learning, is a feed-forward backpropagation network. It takes as input the output coefficients from the first subnet. It outputs, in its M output nodes, a confidence measure ([0,1]) indicating the extent to which the input image corresponds to each of the M distinctive scenes.

In Section 2 we discuss the PCA net, its theory and the algorithm behind it. Section 3 deals with the backpropagation net. We conclude in Section 4 with some observations and comments on future directions that this work suggests.

2. PCA NET

The PCA net is a single layered network of n inputs $X = [x_1, x_2, \ldots, x_n]^T$ and m outputs $Y = [y_1, y_1, \ldots, y_m]^T$. (See Figure 1.)

Each input x_i corresponds to an image pixel. Here,

$$n = (\#\ of\ image\ rows)(\#\ of\ image\ columns).$$

[1]Research Staff Member Emeritus, IBM T.J. Watson Research Center, Yorktown Heights, NY

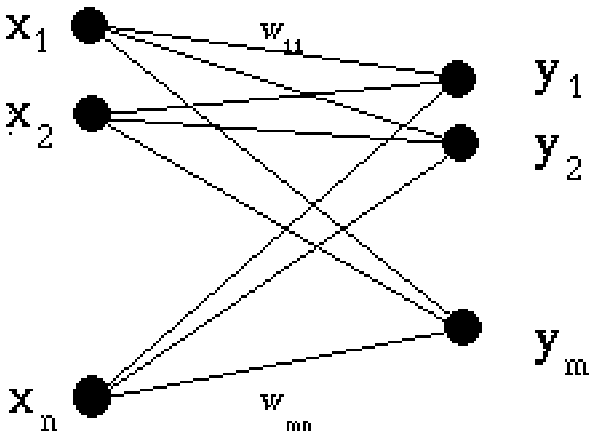

Figure 1: **Diagram of the PCA Net.** The nodes on the left, x_i, receive the gray-level values of the input image pixels. They are connected to output nodes, y_i by weights w_{ij}. The output nodes yield coefficients of the principal components for the given input image.

In our experiment (64x48)-pixel images are used, so that $n = 3072$. We can view the input "vector" by "stacking" columns of an image to form a 3072-component column vector. Since our camera and framegrabber return integral gray-level values between 0 and 255, those are the minimum and maximum values for each x_i.

The strength of the connection between the input x_i and the output y_j is given by a weight w_{ji}. In particular, we have

$$y_j = \sum_i w_{ji} x_i.$$

We use the "Generalized Hebbian Algorithm" (GHA) as the training algorithm. The dynamics specifies the weight of a connection after update (written w_{ij}^{+}): (Sanger, 1989)

$$w_{ij}^{+} = w_{ij} + \gamma(y_i x_j - y_i \sum_{k \leq i} w_{kj} y_k).$$

Setting $W = (w_{ij})$, we express this in matrix form as:

$$\Delta W = \gamma(YX^T - LT[YY^T]W),$$

where $LT[\cdot]$ sets all elements above the diagonal of its matrix argument to zero, thereby making it lower triangular. The learning parameter, γ, specifies the rate of learning and influences how quickly the weights converge and if they converge at all.

We first state, without proof, a convergence theorem. (See Haykin, 1994, Appendix B for a proof.) In the following sections, we explain the related concepts.

Theorem Let the components of W be assigned random values at time zero. Then, with probability 1, W will converge to the matrix whose rows are the first m eigenvectors of the input correlation matrix $Q = E[XX^T]$, ordered by decreasing eigenvalue. (Note that Q is a symmetric matrix.)

2.1 Principal Component Analysis

The motivation behind the GHA algorithm is to compress data and to preserve as much of the information in the input as possible. For example, in our implementation we reduce a 3072-pixel (i.e., 3072-dimension) image into a 4-dimensional vector. Principal Component Analysis allows us to find the four dimensional vector which captures the most variance (which we may view as a measure of information) in the original data.

Let X be a random variable each component of which has zero mean. (A change of variables ensures this in the general case.) Consider the collection of all possible images as the sample space of the random variable. The autocorrelation matrix Q of the input signal distribution is defined by

$$Q = E[XX^T],$$

where E is the expectation operator. Let u_i and λ_i ($i = 1, 2, \ldots, n$) be the orthonormal eigenvectors and the corresponding eigenvalues of Q, respectively, the latter taken in decreasing order. Define the corresponding matrices

$$U = [u_1, u_2, \ldots, u_n]$$

and

$$\Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}.$$

Eigenvectors u_i of Q are called "principal components" in signal processing (hence the name of the net). If the eigenvalues are distinct,

$$Q = U\Lambda U^T.$$

By using u_i as basis vectors, a given image, X, can be expressed as linear combinations of those basis vectors, as follows.

$$X = \sum_{i=1}^{n} u_i y_i = UY.$$

So that, for the coefficient vector, we have

$$Y = U^T X.$$

Since U is unitary, $U^{-1} = U^T$. The coefficient y_i is thus the magnitude of component u_i contained in X.

Put another way, typically there is much redundant information in a raw image. Except for rare cases (e.g., the noise of a "snowy" television channel), the pixels in the image vector, $X = [x_1, x_2, \ldots, x_n]^T$ are "correlated". By this we mean informally that the gray-level values at some points in a picture are predictable from these values at other points in the same picture - all of the pixels in a full moon, for example,

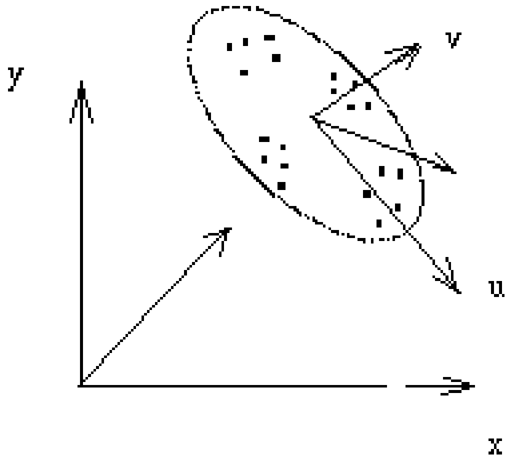

Figure 2: **Principal Components.** The principal components are the semiaxes, u and v, of an ellipse which contains the data points.

will be close to white. To obtain a more compact representation of the information in the image, we seek the above transformation U^T. The latter ensures that $Y = [y_1, y_2, \ldots, y_n]^T$ has *un*correlated components. By a theorem of Karhunen and Loeve (Rosenfeld & Kak, 1981) the transform matrix U^T consists of the eigenvectors of the autocorrelation matrix Q. The weights of the PCA net converge to these eigenvectors, so that the net's output is uncorrelated. Since the redundancies in the inputs are removed, the output variance will be maximized. The outputs represent the largest possible amount of information which a fixed, small number of dimensions (four, in our case) can convey. It might be easier to understand the geometry of principal components through Figure 2.

In the figure xy–space is a n-dimensional vector space. Each point in xy space represents an image (that is, represents the n-dimensional vector which corresponds to the gray-level values of the pixels in an image). The basis for xy–space is orthogonal, but might not represent the information contained in the images as efficiently as possible. As we noted, the projection of each image-as-a-vector onto those basis vectors will be correlated (with redundant information). We seek a new set of basis vectors that are mutually orthogonal and which better represent the variance of the data vectors. We proceed as follows: We find a best-fit n-dimensional ellipse (or ellipsoid) around the vectors. The largest axis of the ellipsoid is the first principal component, the second largest is the second principal component, and so on until the nth largest axis is the nth principal component. We can see that the new basis vectors, as semiaxes of an ellipse, are indeed orthogonal to each other. As it turns out, the new basis vectors are the principal components of the images, which are the same thing as the eigenvectors of the correlation matrix. (Rosenfeld & Kak, 1981)

2.2 Advantages and Disadvantages

From these considerations we see the significance of the GHA algorithm. The dynamics defining ΔW define the principal component vectors in question. Moreover, the vectors need not be computed, as the net itself instantiates the dynamics and the

Figure 3: **Training Images.** We chose five different views of each of five parts of the Vision Lab at Yale. The four most important principal components are extracted and the backpropagation net is trained on the coefficients of those components.

appropriate subsequent pixel processing. For $n = 3000$ input nodes, $Q = E[XX^T]$ has 9 million components. If the number of outputs is much smaller than the number of inputs, as in our case, GHA finds the most important eigenvectors - that is, the eigenvectors with largest eigenvalues without having to decompose the huge matrix Q. In addition, GHA is a neural net algorithm with the potential for high-speed, special-purpose hardware.

A disadvantage is that GHA provides only an approximation to the eigenvectors. Furthermore, as in all such numerical methods, errors in the first few eigenvectors will magnify the errors in the subsequent eigenvectors, so that the algorithm has poor numerical accuracy for all but the first few eigenvectors. For our images, the number of samples, N, is very small compared with n, the dimensionality of the space where the samples are drawn. Therefore, we require only the first few eigenvectors (the principal components). Note, too, that the algorithm only involves local operations. Hence, it is possible to implement GHA on a parallel machine, though the communication of data will require an overhead of time.

2.3 Experiment

The PCA net is trained with $N = 25$ images, five each of five distinctive scenes from Yale's Vision Lab. (See Figure 3). Each image is input to the net 60 times.

GHA is used as the learning rule to adjust the weights of the network. Although normalization of input does not affect the output of the net, it does influence the choice of learning rate γ. (Experiments with power spectra of images as input, for instance, will require a different learning rate.) We take an experimental approach to specifying a good value of γ. Too large a γ will drive the net to saturation, where the values of the connections are outside of the representable range of the computer, and too small a γ will make the net take too long to converge. For our net with 256-

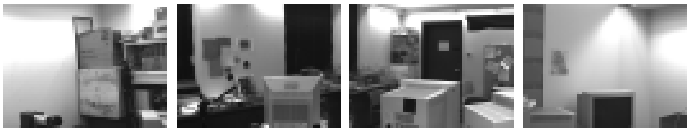

Figure 4: **Test Images.** These are four different views, one of the five views above is omitted. We provided these as inputs to the system after it had been trained on the images in Figure 3.

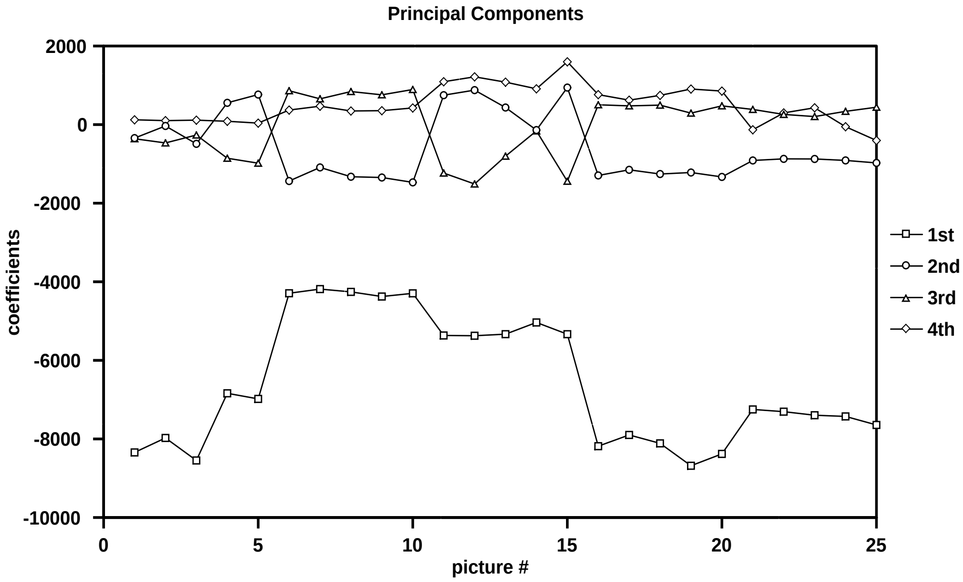

Figure 5: **Coefficients of the Principal Components.** The values of the coefficients of the principal components obtained from the PCA net on the twenty training images. They are grouped in fives: 1-5, 6-10, 11-15, 16-20, and 21-25. Each such group contains five slightly different images of the same scene.

level gray-scale images as input, γ was set to 10^{-8}. Several observations were made while tuning this learning rate. First, γ should decrease with time. Second, different output nodes should use different γ values, since each coefficient (node) converges with slightly different rates. We employed a purely empirical approach to this tuning of the γ values.

Once the net converges, we have found - to some accuracy - the principal component vectors for our data set. Those vectors are stored as the weights of our net. Next, we freeze the weights and input the training set again. For each such input we obtain the coefficients of those principal components, one coefficient at each output node. The coefficients from our images are plotted in Figure 5. These coefficients are the values of the projection of the N vectors (images) onto the first M (in our case M = 4) principal components (the basis in uv-space).

In Figure 5 the first five picture numbers (labeled on the horizontal axis) correspond to the same scene, as do the following five, the third set of five, and the last five. The flatness of each of the five curves shows that images of the same scene have similar coefficients. We represent, then, a location by the coefficients which are characteristic of images taken of that location. Suppose that an arbitrary set of coefficients is specified. We need to decide how much that set of coefficients resembles the sets which have been stored. If the coefficients of a new scene are sufficiently close to one of the stored sets, we will conclude that the new scene is the same as the one in our database. In this way we are able to decide whether or not we are at a previously-seen location, as well as which such location. We decided that, in principle, any such decision requires a teacher. We chose to teach a feedforward net the classifications we wanted; in a sense, the training of that feedforward net will provide the idea of "sameness" that we intuitively feel.

3. BACKPROP NET

We did not apply backpropagation directly to train the feedforward net to identify scenes – the most salient reason being that backprop is slow to converge. With a net of 3072 input nodes there would be

$$3072(\textit{number of hidden nodes}) + (\textit{hidden nodes})(\textit{number of output nodes})$$

free parameters to tune, and we have no clear heuristic to guide the tuning.

In addition, human scene recognition involves considerable preprocessing – edge detection, noise reduction, feature extraction, etc. It is therefore not surprising that a stand-alone backprop net converges slowly, since the complexity of that preprocessing must be expressed in the net's weights. Hence, it is desirable that we separate some preprocessing into a separate system and reduce the number of the free parameters in the backprop net. Biological results have shown that the equivalent of a principal component analyzer exists in receptive fields. (Rubner & Schulten, 1990)

We take it that a place recognition system can not rely solely on the unsupervised learning of the PCA net. Consider, for example, a Hopfield Net acting as a memory, and which does not require supervision or other outside judgment of its performance. In Figure 6, we show that a trained Hopfield net when stimulated by a slight variant of one of its training samples, does not give a correct recall. Rather, it outputs an image that doesn't resemble any of the training samples, a so-called spurious state. It may be that the problem of spurious states does not stem from the architecture of a Hopfield net, but rather from a fundamental lack of information. The ability to recall a "noiseless" version of a "noisy" stimulus is attributed to a Hopfield net. Just what is a "noisy" stimulus? How can it be distinguished from a new "noiseless" stimulus or - even worse - from a new, but different, "noisy" stimulus? The "spurious" state is actually a correct recall, in so far as the Hopfield net is concerned. These questions motivated the conclusion that wholly unsupervised learning is not appropriate for place recognition. We want machines to perform a classification similar to the ones we, as sentient agents, agree on. Some sentient intervention is thus required. We must somehow impart the assumptions which we internalize to any system which we want to act as we do. Simply put, we found it effective to introduce supervised learning in the place recognition problem.

3.1 Experiment

The "art" of creating a backpropagation net lies in choosing how many free parameters we permit the system to learn. If we have too few free parameters (by having too

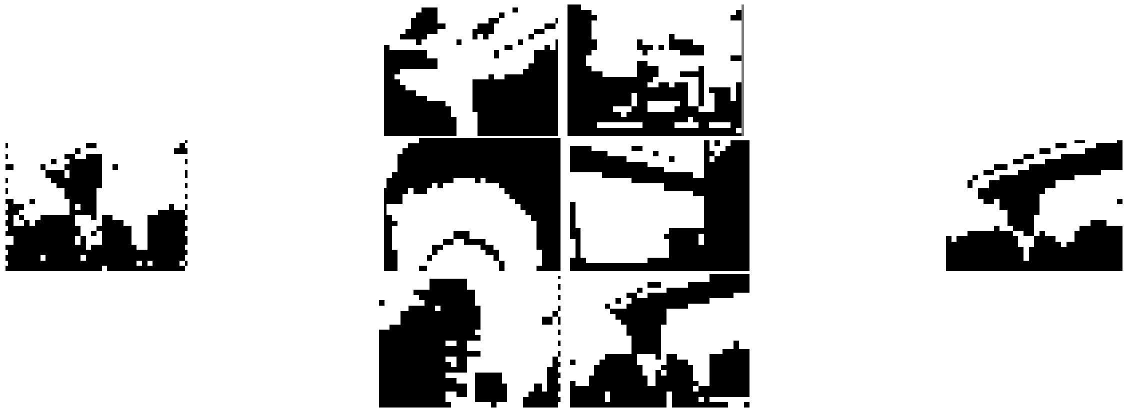

Figure 6: **Hopfield Net.** The middle columns are the training set. All are low-resolution photographs. The left column is a noisy version of the lower right training sample. The right column is the erroneous recall, i.e., a spurious state.

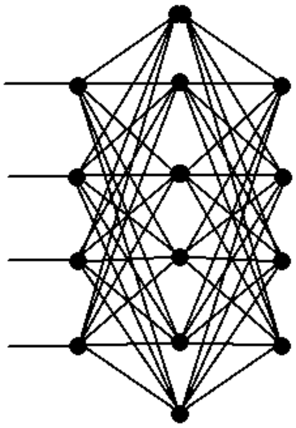

Figure 7: **Backprop Net Schematic.** The left column of neurons receive the inputs from the PCA net. The middle column comprise the hidden nodes. The right column is the five output nodes, one per location.

few hidden nodes) then we will not be able to capture the complexity of the classes. The XOR problem, as a fundamental example, is not solvable by a backprop net without hidden nodes. If we have too many free parameters, the learning will require longer to converge. Worse yet, the net will "overfit" the sample data it receives in its epochs of input. While it will learn the exact input-output mapping we teach it, that mapping will not generalize well to other, unseen inputs. Empirically, we found that a backprop net with 6 hidden nodes sandwiched between 4 inputs (one for each of the four principal component coefficients) and 5 outputs (one for each of the five places) worked well for our problem of recognizing lab scenes and, in addition, converged quickly.

The net was initialized with small, random values for weights between layers. The transfer function is the sigmoidal

$$y_j = \frac{1}{1 + e^{-v_j}},$$

$$v_j = \sum w_{ji} u_i,$$

where y_j is the output of neuron j and u_i is both the output of neuron i and the input to neuron j. w_{ji} is the weight connecting the ith input to neuron j itself. The activation level of neuron j is v_j, which is the weighted sum of the inputs to that neuron.

The inputs are presented to the network in epochs, in each of which all N inputs (N = 25 in our case) are processed. The five output nodes in each case yield values in the interval (0,1). The "correct" value, i.e., the one we want the network to yield is a five-tuple with a 0 in each of the components which corresponds to an incorrect location and a 1 in the component which corresponds to the correct location. Of course "correct" and "incorrect" are determined by human observers. Note that the network can not output the precise expected value, since the range of the sigmoidal transfer function does not contain 0 or 1. Thus, the weights will diverge toward infinity, as they drive the sigmoidal function toward its limiting values. We do not use a step function, since we desire the continuum of output values. The range (0,1) provides a value for interpretation as "confidence" in the recognition. Our network consisted of four inputs, taken from the four outputs of the PCA net. It also contained six nodes in its middle layer and five outputs, one for each of the five scenes the system was meant to recognize. We stopped the training after 20,000 epochs, which required approximately 30 seconds on a Sun Sparc IPX.

When the net outputs a vector, that output is compared to the desired result, and the weights are adjusted in the direction of the local gradient throughout the network – the standard backpropagation algorithm. The learning rate and momentum parameter can be adjusted to speed the convergence of the net and avoid sending the coefficients off to infinity. (We found that a learning rate, $\eta = 0.1$ and a momentum parameter, $\alpha = 0.3$ performed well through experimentation.) The weights w_{ij} are frozen after training to store the classifications the net has just learned. Finally, the unseen test images are input to the net, and the outputs are compared to the desired outputs. The following table yields the results of the five input test images.

No.	*Output 1*	*Output 2*	*Output 3*	*Output 4*	*Output 5*
1	.939	$4.12 * 10^{-4}$	$1.10 * 10^{-2}$	$2.82 * 10^{-4}$	$7.18 * 10^{-2}$
2	$6.96 * 10^{-5}$	.946	$1.91 * 10^{-2}$	$6.67 * 10^{-2}$	$1.38 * 10^{-2}$
3	$3.19 * 10^{-2}$	$3.65 * 10^{-3}$	.979	$2.09 * 10^{-2}$	$1.20 * 10^{-5}$
4	$1.99 * 10^{-4}$	$5.81 * 10^{-2}$	$5.31 * 10^{-2}$	.921	$5.34 * 10^{-2}$

3. OBSERVATIONS AND CONCLUSIONS

It is well-known that the human vision system has a layered structure. There have been attempts (Leen et al, 1992, Dumont & Yang, 1991, Metcalfe & Cottrell, 1991) to decompose the difficult problem of recognition into subproblems and to use different neural networks to solve each subproblem. Our PCA net and backprop net work in a similar fashion, i.e., the PCA net is a preprocessor (Principal Component Analyzer) for the backprop net, which is the classifier. We could also extend this a few more stages. For example, we could have preprocessors to extract features, such as edges. Then, instead of using raw images as input, the PCA net could work on an

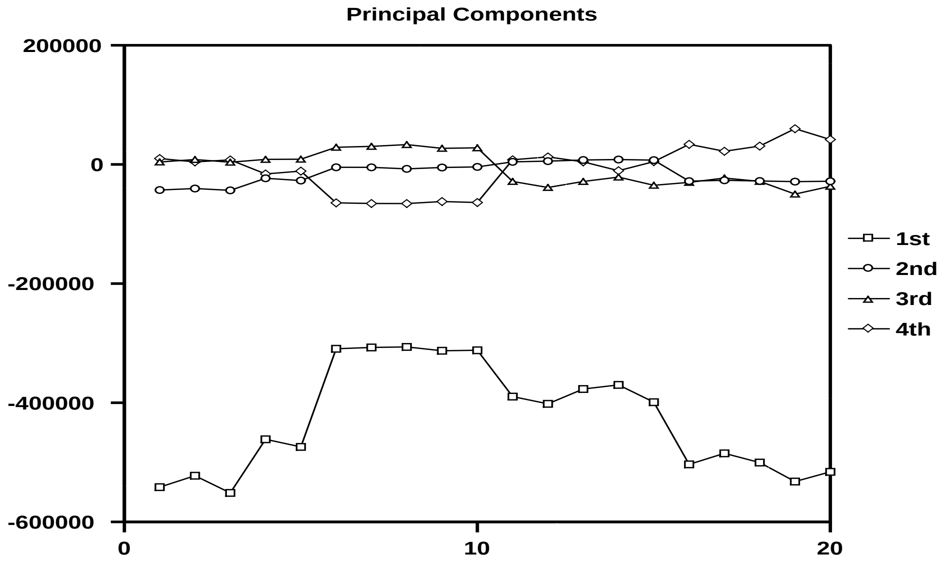

Figure 8: **Coefficients of Principal Components in the Fourier Domain.** This figure is analogous to Figure 5. In this case, the data are the coefficients of PCs when images are described in the Fourier Domain

"edge map," for example, to increase efficiency. Here we show an example where we introduce a Fourier Transform as a preprocessor:

$$raw\ images \overset{Fourier}{\rightarrow} Power\ Spectrum \overset{PCA}{\rightarrow} coefficients \overset{backprop}{\rightarrow} locations$$

If an image rotates by a small angle, it becomes an entirely different image, if considered as a point in 3072-dimensional Cartesian space. The change is relatively small, however, in the Fourier domain. Figure 8 again shows the PCA coefficients of our 25 training samples. In this case, the power spectra of the images were used as input. It would be possible to train the backprop net on the coefficients in Figure 8, rather than those in Figure 5. For this work, we restricted our attention to the simpler approach described in Sections 2 and 3.

Here we sought to solve a problem of general importance - place recognition - with the use of neural nets in general and Hebbian Dynamics in particular. The PCA net, which used Hebbian dynamics to extract the most salient components along which to represent the images, allowed a very simple classifier to handle very complex images well. If we interpret the output of the supervised, backpropagation net as a "confidence" in the system's recognition a certain place, the tests we did (see the table in Section 3) showed that with a confidence greater than .9 the recognizer correctly identified the four test scenes. On the same test scenes, the confidence in the incorrect places (the off diagonal entries) was less than .25. Further work could extend this proof of concept in several directions. For one, larger images would almost certainly need more preprocessing before the PCA net is used to extract the salient features. Other work might investigate the performance of the system as the backprop net is trained with more locations. In our preliminary work presented here, we have shown that with Hebbian dynamics a neural-network-based approach is a viable foundation for a basic location-recognition system.

References

[1] J.Metcalfe G.W.Cottrell. Empath: Face, emotion, and gender recognition using holons. *Advances in Neural Information Processing System 3*, 564 – 571, 1991.

[2] G.A.Dumont J.Yang. Classification of acousitic emission signals via Hebbian feature extraction. *International Joint Conference on Neural Networks*, 113 – 118, 1991.

[3] A. Rosenfeld and A.C.Kak. *Digital Picture Processing*, volume 1. Academic Press, Inc., 2nd edition, 1981.

[4] Jeanne Rubner and K[laus] Schulten. Development of feature detectors by self-organization: A network model. *Biological Cybernetics*, 62:193 – 199, 1990.

[5] Terence D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459 – 473, 1989.

[6] D.Hammerstrom T.K.Leen, M.Rudnick. Hebbian feature discovery improves classifier efficiency. *International Joint Conference on Neural Networks*, 4:158 – 164, 1992.

[7] Simon Haykin. Neural Networks: A Comprehensive Foundation. Macmillan College Publishing Company, New York, 1994.