Neural Network-Based Face Detection*

Henry A. Rowley
har@cs.cmu.edu
http://www.cs.cmu.edu/~har

Shumeet Baluja
baluja@cs.cmu.edu
http://www.cs.cmu.edu/~baluja

Takeo Kanade
tk@cs.cmu.edu
http://www.cs.cmu.edu/~tk

School of Computer Science, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Abstract

We present a neural network-based face detection system. A retinally connected neural network examines small windows of an image, and decides whether each window contains a face. The system arbitrates between multiple networks to improve performance over a single network. We use a bootstrap algorithm for training the networks, which adds false detections into the training set as training progresses. This eliminates the difficult task of manually selecting non-face training examples, which must be chosen to span the entire space of non-face images. Comparisons with other state-of-the-art face detection systems are presented; our system has better performance in terms of detection and false-positive rates.

1 Introduction

In this paper, we present a neural network-based algorithm to detect frontal views of faces in gray-scale images[1]. The algorithms and training methods are general, and can be applied to other views of faces, as well as to similar object and pattern recognition problems.

Training a neural network for the face detection task is challenging because of the difficulty in characterizing prototypical "non-face" images. Unlike face *recognition*, in which the classes to be discriminated are different faces, the two classes to be discriminated in face *detection* are "images containing faces" and "images not containing faces". It is easy to get a representative sample of images which contain faces, but it is much harder to get a representative sample of those which do not. The size of the training set for the second class can grow very quickly.

We avoid the problem of using a huge training set for non-faces by selectively adding images to the training set as training progresses [Sung and Poggio, 1994]. This "bootstrap" method reduces the size of the training set needed. Detailed descriptions of this training method, along with the network architecture are given in Section 2. In Section 3, the performance of the system is examined. We find that the system is able to detect 90.5% of the faces over a test set of 130 images, with an acceptable number of false positives. Section 4 compares this system with similar systems. Conclusions and directions for future research are presented in Section 5.

2 Description of the System

Our system operates in two stages: it first applies a set of neural network-based filters to an image, and then uses an arbitrator to combine the filter outputs. The filter examines each location in the image at several scales, looking for locations that might contain a face. The arbitrator then merges detections

*This work was supported by a grant from Siemens Corporate Research, Inc., by the Army Research Office under grant number DAAH04-94-G-0006, and by the Office of Naval Research under grant number N00014-95-1-0591. This work was started while Shumeet Baluja was supported by a National Science Foundation Graduate Fellowship. He is currently supported by a graduate student fellowship from the National Aeronautics and Space Administration, administered by the Lyndon B. Johnson Space Center. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies or endorsements, either expressed or implied, of the sponsoring agencies.

[1] An interactive demonstration is available on the World Wide Web at http://www.cs.cmu.edu/~har/faces.html, which allows anyone to submit images for processing by the face detector, and displays the detection results for pictures submitted by others.

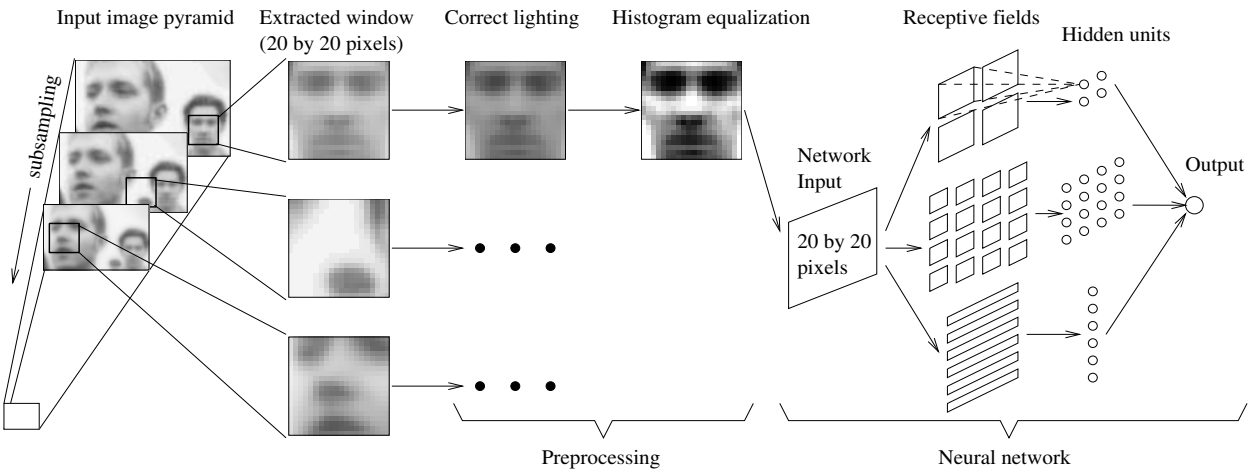

Figure 1: The basic algorithm used for face detection.

from individual filters and eliminates overlapping detections.

2.1 Stage One: A Neural Network-Based Filter

The first component of our system is a filter that receives as input a 20x20 pixel region of the image, and generates an output ranging from 1 to -1, signifying the presence or absence of a face, respectively. To detect faces anywhere in the input, the filter is applied at every location in the image. To detect faces larger than the window size, the input image is repeatedly reduced in size (by subsampling), and the filter is applied at each size. The filter itself must have some invariance to position and scale. The amount of invariance built into the filter determines the number of scales and positions at which the filter must be applied. For the work presented here, we apply the filter at every pixel position in the image, and scale the image down by a factor of 1.2 for each step in the pyramid.

The filtering algorithm is shown in Figure 1. First, a preprocessing step, adapted from [Sung and Poggio, 1994], is applied to a window of the image. The window is then passed through a neural network, which decides whether the window contains a face. The preprocessing first attempts to equalize the intensity values across the window. We fit a function which varies linearly across the window to the intensity values in an oval region inside the window. Pixels outside the oval may represent the background, so those intensity values are ignored in computing the lighting variation across the face. The linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions. Then histogram equalization is performed, which non-linearly maps the intensity values to expand the range of intensities in the window. The histogram is computed for pixels inside an oval region in the window. This compensates for differences in camera input gains, as well as improving contrast in some cases.

The preprocessed window is then passed through a neural network. The network has retinal connections to its input layer; the receptive fields of hidden units are shown in Figure 1. There are three types of hidden units: 4 which look at 10x10 pixel subregions, 16 which look at 5x5 pixel subregions, and 6 which look at overlapping 20x5 pixel horizontal stripes of pixels. Each of these types was chosen to allow the hidden units to represent features that might be important for face detection. In particular, the horizontal stripes allow the hidden units to detect such features as mouths or pairs of eyes, while the hidden units with square receptive fields might detect features such as individual eyes, the nose, or corners of the mouth. Although the figure shows a single hidden unit for each subregion of the input, these units can be replicated. For the experiments which are described later, we use networks with two and three sets of these hidden units. Similar input connection patterns are commonly used in speech and character recognition tasks [Waibel *et al.*, 1989, Le Cun *et al.*, 1989]. The network has a single,

real-valued output, which indicates whether or not the window contains a face.

Examples of output from a single network are shown in Figure 2. In the figure, each box represents the position and size of a window to which the neural network gave a positive response. The network has some invariance to position and scale, which results in multiple boxes around some faces. Note also that there are some false detections; they will be eliminated by methods presented in Section 2.2.

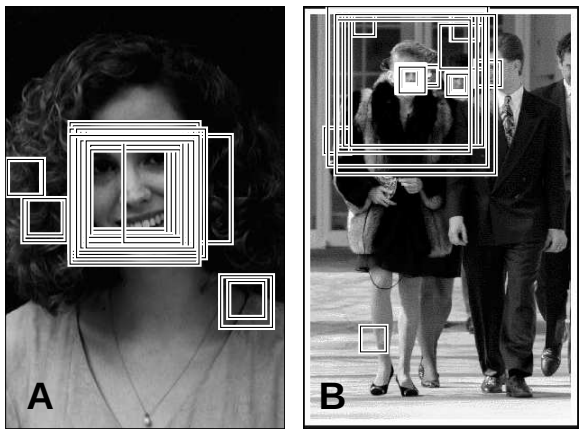

Figure 2: Images with all the above threshold detections indicated by boxes.

To train the neural network used in stage one to serve as an accurate filter, a large number of face and non-face images are needed. Nearly 1050 face examples were gathered from face databases at CMU and Harvard[2]. The images contained faces of various sizes, orientations, positions, and intensities. The eyes and the center of the upper lip of each face were located manually, and these points were used to normalize each face to the same scale, orientation, and position, as follows:

1. The image is rotated so that both eyes appear on a horizontal line.
2. The image is scaled so that the distance from the point between the eyes to the upper lip is 12 pixels.
3. A 20x20 pixel region, centered 1 pixel above the point between the eyes and the upper lip, is extracted.

[2]Dr. Woodward Yang at Harvard provided over 400 mugshot images which we used for training.

In the training set, 15 face examples are generated from each original image, by randomly rotating the images (about their center points) up to 10°, scaling between 90% and 110%, translating up to half a pixel, and mirroring. Each 20x20 window in the set is then preprocessed (by applying lighting correction and histogram equalization). A few example images are shown in Figure 3. The randomization gives the filter invariance to translations of less than a pixel and scalings of $\pm 10\%$. Larger changes in translation and scale are dealt with by applying the filter at every pixel position in an image pyramid, in which the images are scaled by factors of 1.2.

Figure 3: Example face images, randomly mirrored, rotated, translated, and scaled by small amounts.

Practically any image can serve as a non-face example because the space of non-face images is much larger than the space of face images. However, collecting a "representative" set of non-faces is difficult. Instead of collecting the images before training is started, the images are collected during training, in the following manner, adapted from [Sung and Poggio, 1994]:

1. Create an initial set of non-face images by generating 1000 images with random pixel intensities. Apply the preprocessing steps to each of these images.
2. Train a neural network to produce an output of 1 for the face examples, and -1 for the non-face examples. The training algorithm is standard error backpropogation. On the first iteration of this loop, the network's weights are initially random. After the first iteration, we use the

weights computed by training in the previous iteration as the starting point for training.

3. Run the system on an image of scenery *which contains no faces*. Collect subimages in which the network incorrectly identifies a face (an output activation > 0).
4. Select up to 250 of these subimages at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

Some examples of non-faces that are collected during training are shown in Figure 4. We used 120 images of scenery for collecting negative examples in this bootstrap manner. A typical training run selects approximately 8000 non-face images from the 146,212,178 subimages that are available at all locations and scales in the training scenery images.

2.2 Stage Two: Merging Overlapping Detections and Arbitration

The examples in Figure 2 showed that the raw output from a single network will contain a number of false detections. In this section, we present two strategies to improve the reliability of the detector: merging overlapping detections from a single network and arbitrating among multiple networks.

2.2.1 Merging Overlapping Detections

Note that in Figure 2, most faces are detected at multiple nearby positions or scales, while false detections often occur with less consistency. This observation leads to a heuristic which can eliminate many false detections. For each location and scale at which a face is detected, the number of detections within a specified neighborhood of that location can be counted. If the number is above a threshold, then that location is classified as a face. The centroid of the nearby detections defines the location of the detection result, thereby collapsing multiple detections. In the experiments section, this heuristic will be referred to as "thresholding".

If a particular location is correctly identified as a face, then all other detection locations which overlap it are likely to be errors, and can therefore be eliminated. Based on the above heuristic regarding nearby detections, we preserve the location with the higher number of detections within a small neighborhood, and eliminate locations with fewer detections. Later, in the discussion of the experiments, this heuristic is called "overlap elimination". There are relatively few cases in which this heuristic fails; however, one such case is illustrated in the left two faces in Figure 2B, in which one face partially occludes another.

The implementation of these two heuristics is as follows. Each detection by the network at a particular location and scale is marked in an image pyramid, labelled the "output" pyramid. Then, each location in the pyramid is replaced by the number of detections in a specified neighborhood of that location. This has the effect of "spreading out" the detections. The neighborhood extends an equal number of pixels in the dimensions of scale and position. A threshold is applied to these values, and the centroids (in both position and scale) of all above threshold regions are computed. All detections contributing to the centroids are collapsed down to single points. Each centroid is then examined in order, starting from the ones which had the highest number of detections within the specified neighborhood. If any other centroid locations represent a face overlapping with the current centroid, they are removed from the output pyramid. All remaining centroid locations constitute the final detection result.

2.2.2 Arbitration among Multiple Networks

To further reduce the number of false positives, we can apply multiple networks, and arbitrate between the outputs to produce the final decision. Each network is trained in the manner described above, but with different random initial weights, random initial non-face images, and random permutations of the order of presentation of the scenery images. As will be seen in the next section, the detection and false positive rates of the individual networks will be quite close. However, because of different training conditions and because of self-selection of negative training examples, the networks will have different biases and will make different errors.

Each detection by a network at a particular position and scale is recorded in an image pyramid. One

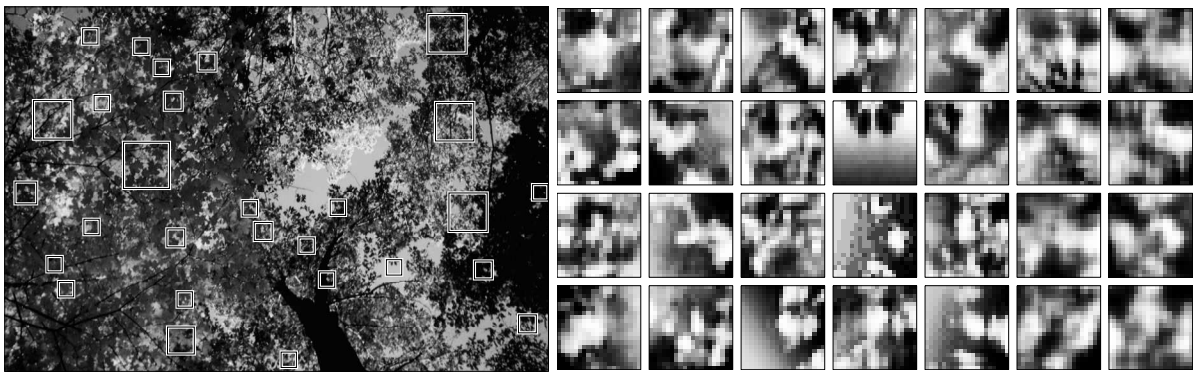

Figure 4: During training, the partially-trained system is applied to images of scenery which do not contain faces (like the one on the left). Any regions in the image detected as faces (which are expanded and shown on the right) are errors, which can be added into the set of negative training examples.

way to combine two such pyramids is by ANDing them. This strategy signals a detection only if both networks detect a face at precisely the same scale and position. Due to the biases of the individual networks, they will rarely agree on a false detection of a face. This allows ANDing to eliminate most false detections. Unfortunately, this heuristic can decrease the detection rate because a face detected by only one network will be thrown out. However, we will show later that individual networks can all detect roughly the same set of faces, so that the number of faces lost due to ANDing is small.

Similar heuristics, such as ORing the outputs of two networks, or voting among three networks, were also tried. Each of these arbitration methods can be applied before or after the "thresholding" and "overlap elimination" heuristics. If applied afterwards, we combine the centroid locations rather than actual detection locations, and require them to be within some neighborhood of one another rather than precisely aligned.

Arbitration strategies such as ANDing, ORing, or voting seem intuitively reasonable, but perhaps there are some less obvious heuristics that could perform better. In [Rowley *et al.*, 1995], we tested this hypothesis by using a separate neural network to arbitrate among multiple detection networks. It was found that the neural network-based arbitration produces results comparable to those produced by the heuristics presented earlier.

3 Experimental Results

A large number of experiments were performed to evaluate the system. We first show an analysis of which features the neural network is using to detect faces, then present the error rates of the system over three large test sets.

3.1 Sensitivity Analysis

In order to determine which part of the input image the network uses to decide whether the input is a face, we performed a sensitivity analysis using the method of [Baluja and Pomerleau, 1995]. We collected a positive test set based on the training database of face images, but with different randomized scales, translations, and rotations than were used for training. The negative test set was built from a set of negative examples collected during the training of an earlier version of the system. Each of the 20x20 pixel input images was divided into 100 2x2 pixel subimages. For each subimage in turn, we went through the test set, replacing that subimage with random noise, and tested the neural network. The resulting sum of squared errors made by the network is an indication of how important that portion of the image is for the detection task. Plots of the error rates for two networks we developed are shown in Figure 5. Network 1 uses two sets of the hidden units illustrated in Figure 1, while Network 2 uses three sets.

The networks rely most heavily on the eyes, then on the nose, and then on the mouth (Figure 5). Anecdotally, we have seen this behavior on several real

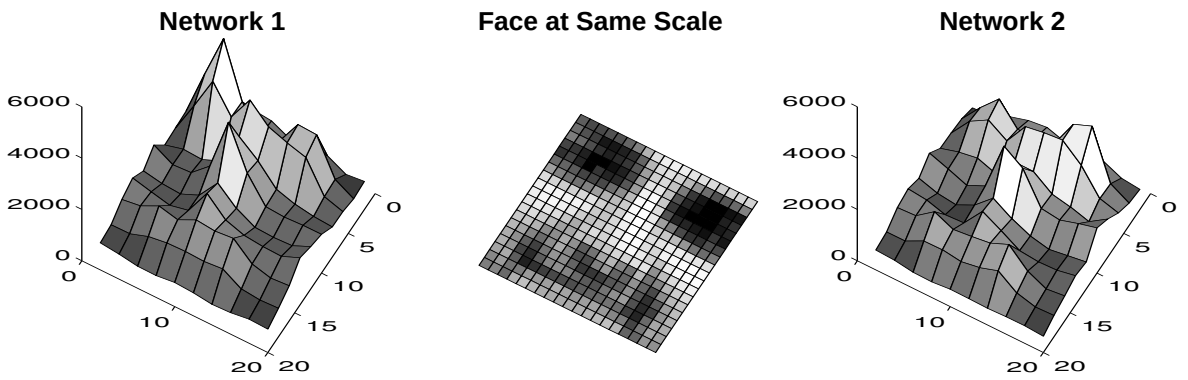

Figure 5: Error rates (vertical axis) on a small test resulting from adding noise to various portions of the input image (horizontal plane), for two networks. Network 1 has two copies of the hidden units shown in Figure 1 (a total of 58 hidden units and 2905 connections), while Network 2 has three copies (a total of 78 hidden units and 4357 connections).

test images. Even in cases in which only one eye is visible, detection of a face is possible, though less reliable, than when the entire face is visible. The system is less sensitive to the occlusion of other features such as the nose or mouth.

3.2 Testing

The system was tested on three large sets of images, which are completely distinct from the training sets. Test Set A was collected at CMU, and consists of 42 scanned photographs, newspaper pictures, images collected from the World Wide Web, and digitized television pictures. These images contain 169 frontal views of faces, and require the networks to examine 22,053,124 20x20 pixel windows. Test Set B consists of 23 images containing 155 faces (9,678,084 windows); it was used in [Sung and Poggio, 1994] to measure the accuracy of their system. Test Set C is similar to Test Set A, but contains some images with more complex backgrounds and without any faces, to more accurately measure the false detection rate. It contains 65 images, 183 faces, and 51,368,003 windows.[3]

A feature our face detection system has in common with many systems is that the outputs are not binary. The neural network filters produce real values between 1 and -1, indicating whether or not the input contains a face, respectively. A threshold value of zero is used during *training* to select the negative examples (if the network outputs a value of greater than zero for any input from a scenery image, it is considered a mistake). Although this value is intuitively reasonable, by changing this value during *testing*, we can vary how conservative the system is. To examine the effect of this threshold value during testing, we measured the detection and false positive rates as the threshold was varied from 1 to -1. At a threshold of 1, the false detection rate is zero, but no faces are detected. As the threshold is decreased, the number of correct detections will increase, but so will the number of false detections. This tradeoff is illustrated in Figure 6, which shows the detection rate plotted against the number of false positives as the threshold is varied, for the two networks presented in the previous section. Since the zero threshold locations are close to the "knees" of the curves, as can be seen from the figure, we used a zero threshold value throughout testing. Experiments are currently underway to examine the effect of the threshold value used during training.

Table 1 shows the performance for four networks working alone, the effect of overlap elimination and collapsing multiple detections, and the results of using ANDing, ORing, voting, and neural network arbitration. Networks 3 and 4 are identical to Networks 1 and 2, respectively, except that the negative example images were presented in a different order during training. The results for ANDing and ORing

[3]Test Sets A, B, and C are available over the World Wide Web, at the URL http://www.cs.cmu.edu/~har/faces.html.

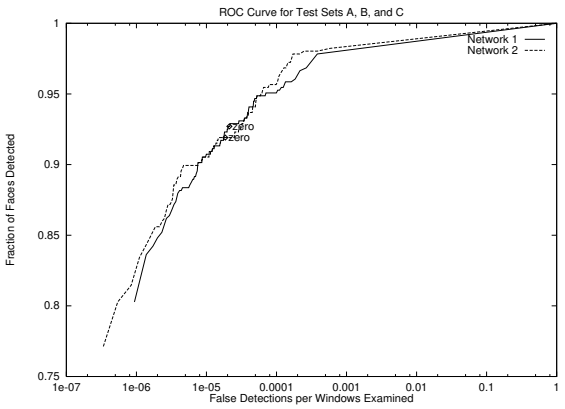

Figure 6: The detection rate plotted against false positives as the detection threshold is varied from -1 to 1, for two networks. The performance was measured over all images from Test Sets A, B, and C. Network 1 uses two sets of the hidden units illustrated in Figure 1, while Network 2 uses three sets. The points labelled "zero" are the zero threshold points which are used for all other experiments.

networks were based on Networks 1 and 2, while voting was based on Networks 1, 2, and 3. The table shows the percentage of faces correctly detected, and the number of false detections over the combination of Test Sets A, B, and C. [Rowley *et al.*, 1995] gives a breakdown of the performance of each of these system for each of the three test sets, as well as the performance of systems using neural networks to arbitration among multiple detection networks.

As discussed earlier, the "thresholding" heuristic for merging detections requires two parameters, which specify the size of the neighborhood used in searching for nearby detections, and the threshold on the number of detections that must be found in that neighborhood. In Table 1, these two parameters are shown in parentheses after the word "threshold". Similarly, the ANDing, ORing, and voting arbitration methods have a parameter specifying how close two detections (or detection centroids) must be in order to be counted as identical.

Systems 1 through 4 show the raw performance of the networks. Systems 5 through 8 use the same networks, but include the thresholding and overlap elimination steps which decrease the number of false detections significantly, at the expense of a small decrease in the detection rate. The remaining systems all use arbitration among multiple networks. Using arbitration further reduces the false positive rate, and in some cases increases the detection rate slightly. Note that for systems using arbitration, the ratio of false detections to windows examined is extremely low, ranging from 1 false detection per 229,556 windows to down to 1 in 10,387,401, depending on the type of arbitration used. Systems 10, 11, and 12 show that the detector can be tuned to make it more or less conservative. System 10, which uses ANDing, gives an extremely small number of false positives, and has a detection rate of about 78.9%. On the other hand, System 12, which is based on ORing, has a higher detection rate of 90.5% but also has a larger number of false detections. System 11 provides a compromise between the two. The differences in performance of these systems can be understood by considering the arbitration strategy. When using ANDing, a false detection made by only one network is suppressed, leading to a lower false positive rate. On the other hand, when ORing is used, faces detected correctly by only one network will be preserved, improving the detection rate. System 13, which uses voting among three networks, yields about the same detection rate and lower false positive rate than System 12, which uses ORing of two networks.

Based on the results shown in Table 1, we concluded that System 11 makes an acceptable tradeoff between the number of false detections and the detection rate. System 11 detects on average 85.4% of the faces, with an average of one false detection per 1,319,035 20x20 pixel windows examined. Figure 7 shows examples output images from System 11.

4 Comparison to Other Systems

[Sung and Poggio, 1994] reports a face detection system based on clustering techniques. Their system, like ours, passes a small window over all portions of the image, and determines whether a face exists in each window. Their system uses a supervised clustering method with six "face" and six "non-face" clusters. Two distance metrics measure the distance of an input image to the prototype clusters. The first metric measures the "partial" distance between the test pattern and the cluster's 75 most significant eigenvectors. The second distance metric is the Euclidean distance between the test pattern

Table 1: Combined Detection and Error Rates for Test Sets A, B, and C

Type	System	Missed faces	Detect rate	False detects	False detect rate
	0) Ideal System	0/507	100.0%	0	0 in 83099211
Single network, no heuristics	1) Network 1 (2 copies of hidden units (52 total), 2905 connections)	37	92.7%	1768	1 in 47002
	2) Network 2 (3 copies of hidden units (78 total), 4357 connections)	41	91.9%	1546	1 in 53751
	3) Network 3 (2 copies of hidden units (52 total), 2905 connections)	44	91.3%	2176	1 in 38189
	4) Network 4 (3 copies of hidden units (78 total), 4357 connections)	37	92.7%	2508	1 in 33134
Single network, with heuristics	5) Network 1 $\rightarrow$ threshold(2,1) $\rightarrow$ overlap elimination	46	90.9%	844	1 in 98459
	6) Network 2 $\rightarrow$ threshold(2,1) $\rightarrow$ overlap elimination	53	89.5%	719	1 in 115576
	7) Network 3 $\rightarrow$ threshold(2,1) $\rightarrow$ overlap elimination	53	89.5%	975	1 in 85230
	8) Network 4 $\rightarrow$ threshold(2,1) $\rightarrow$ overlap elimination	47	90.7%	1052	1 in 78992
Arbitrating among two networks	9) Networks 1 and 2 $\rightarrow$ AND(0)	66	87.0%	209	1 in 397604
	10) Networks 1 and 2 $\rightarrow$ AND(0) $\rightarrow$ threshold(2,3) $\rightarrow$ overlap elimination	107	78.9%	8	1 in 10387401
	11) Networks 1 and 2 $\rightarrow$ threshold(2,2) $\rightarrow$ overlap elimination $\rightarrow$ AND(2)	74	85.4%	63	1 in 1319035
	12) Networks 1 and 2 $\rightarrow$ thresh(2,2) $\rightarrow$ overlap $\rightarrow$ OR(2) $\rightarrow$ thresh(2,1) $\rightarrow$ overlap	48	90.5%	362	1 in 229556
Three nets	13) Networks 1, 2, 3 $\rightarrow$ voting(0) $\rightarrow$ overlap elimination	53	89.5%	195	1 in 426150

threshold(distance,threshold): Only accept a detection if there are at least *threshold* detections within a cube (extending along x, y, and scale) in the detection pyramid surrounding the detection. The size of the cube is determined by *distance*, which is the number of a pixels from the center of the cube to its edge (in either position or scale).

overlap elimination: It is possible that a set of detections erroneously indicate that faces are overlapping with one another. This heuristic examines detections in order (from those having the most votes within a small neighborhood to those having the least), and removing conflicting overlaps as it goes.

voting(distance), AND(distance), OR(distance): These heuristics are used for arbitrating among multiple networks. They take a *distance* parameter, similar to that used by the threshold heuristic, which indicates how close detections from individual networks must be to one another to be counted as occuring at the same location and scale. A *distance* of zero indicates that the detections must occur at precisely the same location and scale. Voting requires two out of three networks to detect a face, AND requires two out of two, and OR requires one out of two to signal a detection.

network arbitration(architecture): The results from three detection networks are fed into an arbitration network. The parameter specifies the network architecture used: a simple perceptron, a network with a hidden layer of 5 fully connected hidden units, or a network with two hidden layers of 5 fully connected hidden units each, with additional connections from the first hidden layer to the output.

Figure 7: Output obtained from System 11 in Table 1. For each image, three numbers are shown: the number of faces in the image, the number of faces detected correctly, and the number of false detections. Some notes on specific images: False detections are present in A and J. Faces are missed in G (babies with fingers in their mouths are not well represented in the training set), I (one because of the lighting, causing one side of the face to contain no information, and one because of the bright band over the eyes), and J (removed because a false detect overlapped it). Although the system was trained only on real faces, hand drawn faces are detected in D. Images A, I, and K were obtained from the World Wide Web, B was scanned from a photograph, C is a digitized television image, D, E, F, H, and J were provided by Sung and Poggio at MIT, G and L were scanned from newspapers, and M was scanned from a printed photograph.

and its projection in the 75 dimensional subspace. These distance measures have close ties with Principal Components Analysis (PCA), as described in [Sung and Poggio, 1994]. The last step in their system is to use either a perceptron or a neural network with a hidden layer, trained to classify points using the two distances to each of the clusters (a total of 24 inputs). Their system is trained with 4000 positive examples and nearly 47500 negative examples collected in the "bootstrap" manner. In comparison, our system uses approximately 16000 positive examples and 9000 negative examples.

Table 2 shows the accuracy of their system on Test Set B, along with the results of our system using the heuristics employed by Systems 10, 11, and 12 in Table 1. In [Sung and Poggio, 1994], 149 faces were labelled in the test set, while we labelled 155. Some of these faces are difficult for either system to detect. Based on the assumption that [Sung and Poggio, 1994] were unable to detect any of the six additional faces we labelled, the number of missed faces is six more than the values listed in their paper. It should be noted that because of implementation details, [Sung and Poggio, 1994] process a slightly smaller number of windows over the entire test set; this is taken into account when computing the false detection rates. Table 2 shows that for equal numbers of false detections, we can achieve higher detection rates.

The main computational cost in [Sung and Poggio, 1994] is in computing the two distance measures from each new window to 12 clusters. We estimate that this computation requires fifty times as many floating point operations as are needed to classify a window in our system, in which the main costs are in preprocessing and applying neural networks to the window.

Although there is insufficient space to present them here, [Rowley *et al.*, 1995] describes techniques for speeding up our system, based on the work of [Umezaki, 1995] on license plate detection. These techniques are related, at a high level, to those presented in [Vaillant *et al.*, 1994]. In that work, two networks were used. The first network has a single output, and like our system it is trained to produce a maximal positive value for centered faces, and a maximal negative value for non-faces. Unlike our system, for faces that are not perfectly centered, the network is trained to produce an intermediate value related to how far off-center the face is. This network scans over the image to produce candidate face locations. It runs quickly because of the network architecture: using retinal connections and shared weights, much of the computation required for one application of the detector can be reused at the adjacent pixel position. This optimization requires any preprocessing to have a restricted form, such that it takes as input the entire image, and produces as output a new image. The window-by-window preprocessing used in our system cannot be used. A second network is used for precise localization: it is trained to produce a positive response for an exactly centered face, and a negative response for faces which are not centered. It is not trained at all on non-faces. All candidates which produce a positive response from the second network are output as detections. A potential problem in [Vaillant *et al.*, 1994] is that the negative training examples are selected manually from a small set of images (indoor scenes, similar to those used for testing the system). It may be possible to make the detectors more robust using the bootstrap technique described here and in [Sung and Poggio, 1994].

5 Conclusions and Future Research

Our algorithm can detect between 78.9% and 90.5% of faces in a set of 130 total images, with an acceptable number of false detections. Depending on the application, the system can be made more or less conservative by varying the arbitration heuristics or thresholds used. The system has been tested on a wide variety of images, with many faces and unconstrained backgrounds.

There are a number of directions for future work. The main limitation of the current system is that it only detects upright faces looking at the camera. Separate versions of the system could be trained for different head orientations, and the results could be combined using arbitration methods similar to those presented here.

Other methods of improving system performance include obtaining more positive examples for training, or applying more sophisticated image preprocessing and normalization techniques. For instance, the

Table 2: Comparison of [Sung and Poggio, 1994] and Our System on Test Set B

System	Missed faces	Detect rate	False detects	False detect rate
10) Networks 1 and 2 → AND(0) → threshold(2,3) → overlap elimination	34	78.1%	3	1 in 3226028
11) Networks 1 and 2 → threshold(2,2) → overlap elimination → AND(2)	20	87.1%	15	1 in 645206
12) Networks 1 and 2 → threshold(2,2) → overlap elim → OR(2) → threshold(2,1) → overlap elimination	11	92.9%	64	1 in 151220
[Sung and Poggio, 1994] (Multi-layer network)	36	76.8%	5	1 in 1929655
[Sung and Poggio, 1994] (Perceptron)	28	81.9%	13	1 in 742175

color segmentation method used in [Hunke, 1994] for color-based face tracking could be used to filter images. The face detector would then be applied only to portions of the image which contain skin color, which would speed up the algorithm as well as eliminating false detections.

One application of this work is in the area of media technology. Every year, improved technology provides cheaper and more efficient ways of storing information. However, automatic high-level classification of the information content is very limited; this is a bottleneck that prevents media technology from reaching its full potential. The work described above allows a user to make queries of the form "Which scenes in this video contain human faces?" and to have the query answered automatically.

Acknowledgements

The authors would like to thank to Kah-Kay Sung and Dr. Tomaso Poggio (at MIT) and Dr. Woodward Yang (at Harvard) for providing a series of test images and a mug-shot database, respectively. Michael Smith (at CMU) provided some digitized television images for testing purposes. We also thank Eugene Fink, Xue-Mei Wang, Hao-Chi Wong, Tim Rowley, and Kaari Flagstad for comments on drafts of this paper.

References

[Baluja and Pomerleau, 1995] Shumeet Baluja and Dean Pomerleau. Encouraging distributed input reliance in spatially constrained artificial neural networks: Applications to visual scene analysis and control. Submitted, 1995.

[Hunke, 1994] H. Martin Hunke. Locating and tracking of human faces with neural networks. Master's thesis, University of Karlsruhe, 1994.

[Le Cun *et al.*, 1989] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropogation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[Rowley *et al.*, 1995] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. CMU-CS-95-158R, Carnegie Mellon University, November 1995. Also available at http://www.cs.cmu.edu/~har/faces.html.

[Sung and Poggio, 1994] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. A.I. Memo 1521, CBCL Paper 112, MIT, December 1994.

[Umezaki, 1995] Tazio Umezaki. Personal communication, 1995.

[Vaillant *et al.*, 1994] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings on Vision, Image, and Signal Processing*, 141(4), August 1994.

[Waibel *et al.*, 1989] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *Readings in Speech Recognition*, pages 393–404, 1989.