Neural Network-Based Face Detection

Henry A. Rowley, Shumeet Baluja, and Takeo Kanade *

Abstract

We present a neural network-based upright frontal 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 present a straightforward procedure for aligning positive face examples for training. To collect negative examples, we use a bootstrap algorithm, which adds false detections into the training set as training progresses. This eliminates the difficult task of manually selecting nonface training examples, which must be chosen to span the entire space of nonface images. Simple heuristics, such as using the fact that faces rarely overlap in images, can further improve the accuracy. Comparisons with several other state-of-the-art face detection systems are presented; showing that our system has comparable performance in terms of detection and false-positive rates.

Keywords: Face detection, Pattern recognition, Computer vision, Artificial neural networks, Machine learning

1 Introduction

In this paper, we present a neural network-based algorithm to detect upright, frontal views of faces in gray-scale images[1]. The algorithm works by applying one or more neural networks directly to portions of the input image, and arbitrating their results. Each network is trained to output the presence or absence of a face. The algorithms and training methods are designed to be general, with little customization for faces.

Many face detection researchers have used the idea that facial images can be characterized directly in terms of pixel intensities. These images can be characterized by probabilistic models of the set of face images [4, 13, 15], or implicitly by neural networks or other mechanisms [3, 12, 14, 19, 21, 23, 25, 26]. The parameters for these models are adjusted either automatically from example images (as in our work) or by hand. A few authors have taken the approach of extracting features and applying either manually or automatically generated rules for evaluating these features [7, 11].

Training a neural network for the face detection task is challenging because of the difficulty in characterizing prototypical "nonface" 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 much harder to get a representative sample of those which do not. We avoid the problem of using a huge training set for nonfaces by selectively adding images to the training set as training progresses [21]. This "bootstrap" method reduces the size of the training set needed. The use of arbitration between multiple networks and heuristics to clean up the results significantly improves the accuracy of the detector.

Detailed descriptions of the example collection and training methods, network architecture, and arbitration methods 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 complex images, with an acceptable number of false positives. Section 4 briefly discusses some techniques that can be used to make the system run faster, and Section 5 compares this system with similar systems. Conclusions and directions for future research are presented in Section 6.

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 outputs. The filters examine each location in the image at several scales, looking for locations that might contain a face. The arbitrator then merges detections 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. This filter must have some invariance to position and scale. The amount of invariance determines the number of scales and positions at which it 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 Fig. 1. First, a preprocessing step, adapted from [21], 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 in 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 (shown in Fig. 2a) 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 preprocessing steps are shown in Fig. 2.

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 Fig. 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 detect local 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 [10, 24]. 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 Fig. 3. 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.

To train the neural network used in stage one to serve as an accurate filter, a large number of face and nonface images are needed. Nearly 1050 face examples were gathered from face databases at CMU, Harvard[2], and from the World Wide Web. The images contained faces of various sizes, orientations, positions, and intensities. The eyes, tip of nose, and corners and center of the mouth of each face were labelled manually. These points were used to normalize each face to the same scale, orientation, and position, as follows:

1. Initialize $\bar{F}$, a vector which will be the average positions of each labelled feature over all the faces, with the feature locations in the first face F_1.

2. The feature coordinates in $\bar{F}$ are rotated, translated, and scaled, so that the average locations of the eyes will appear at predetermined locations in a 20x20 pixel window.

3. For each face i, compute the best rotation, translation, and scaling to align the face's features F_i with the average feature locations $\bar{F}$. Such transformations can be written as a linear function of their parameters. Thus, we can write a system of linear equations mapping the features from F_i to $\bar{F}$. The least squares solution to this over-constrained system yields the parameters for the best alignment transformation. Call the aligned feature locations F'_i.

4. Update $\bar{F}$ by averaging the aligned feature locations F'_i for each face i.

5. Go to step 2.

The alignment algorithm converges within five iterations, yielding for each face a function which maps that face to a 20x20 pixel window. Fifteen face examples are generated for the training set 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 Fig. 4. The randomization gives the filter invariance to translations of less than a pixel and scalings of 20%. 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.

Practically any image can serve as a nonface example because the space of nonface images is much larger than the space of face images. However, collecting a "representative" set of nonfaces is difficult. Instead of collecting the images before training is started, the images are collected during training, in the following manner, adapted from [21]:

1. Create an initial set of nonface images by generating 1000 random images. 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 nonface examples. The training algorithm is standard error backpropogation with momentum [8]. On the first iteration of this loop, the network's weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point.

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 nonfaces that are collected during training are shown in Fig. 5. Note that some of the examples resemble faces, although they are not very close to the positive examples shown in Fig. 4. The presence of these examples forces the neural network to learn the precise boundary between face and nonface images. We used 120 images of scenery for collecting negative examples in the bootstrap manner described above. A typical training run selects approximately 8000 nonface images from the 146,212,178 subimages that are available at all locations and scales in the training scenery images. A similar training algorithm was described in [5], where at each iteration an entirely new network was trained with the examples on which the previous networks had made mistakes.

2.2 Stage Two: Merging Overlapping Detections and Arbitration

The examples in Fig. 3 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 Fig. 3, 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, 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. 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 by the left two faces in Fig. 3B, where one face partially occludes another.

The implementation of these two heuristics is illustrated in Fig. 6. Each detection 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. Normally, the neighborhood extends an equal number of pixels in the dimensions of scale and position, but for clarity in Fig. 6 detections are only spread out in 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 a centroid are collapsed down to a single point. 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. In the face detection work described in [3], similar observations about the nature of the outputs were made, resulting in the development of heuristics similar to those described above.

2.2.2 Arbitration among Multiple Networks

To further reduce the number of false positives, we can apply multiple networks, and arbitrate between their outputs to produce the final decision. Each network is trained in a similar manner, but with random initial weights, random initial nonface images, and 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.

The implementation of arbitration is illustrated in Fig. 7. Each detection at a particular position and scale is recorded in an image pyramid, as was done with the previous heuristics. One 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 different 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 see 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. To test this hypothesis, we applied a separate neural network to arbitrate among multiple detection networks. For a location of interest, the arbitration network examines a small neighborhood surrounding that location in the output pyramid of each individual network. For each pyramid, we count the number of detections in a 3x3 pixel region at each of three scales around the location of interest, resulting in three numbers for each detector, which are fed to the arbitration network, as shown in Fig. 8. The arbitration network is trained to produce a positive output for a given set of inputs only if that location contains a face, and to produce a negative output for locations without a face. As will be seen in the next section, using an arbitration network in this fashion produced results comparable to (and in some cases, slightly better than) those produced by the heuristics presented earlier.

3 Experimental Results

A 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 two large test sets.

3.1 Sensitivity Analysis

In order to determine which part of its input image the network uses to decide whether the input is a face, we performed a sensitivity analysis using the method of [2]. 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 other networks. 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 root mean square error of the network on the test set 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 trained are shown in Fig. 9. Network 1 uses two sets of the hidden units illustrated in Fig. 1, while Network 2 uses three sets.

The networks rely most heavily on the eyes, then on the nose, and then on the mouth (Fig. 9). Anecdotally, we have seen this behavior on several real test images. 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 the nose or mouth.

3.2 Testing

The system was tested on two large sets of images, which are distinct from the training sets. Test Set 1 consists of a total of 130 images collected at CMU, including images from the World Wide Web, scanned from photographs and newspaper pictures, and digitized from broadcast television[3]. It also includes 23 images used in [21] to measure the accuracy of their system. The images contain a total of 507 frontal faces, and require the networks to examine 83,099,211 20x20 pixel windows. The images have a wide variety of complex backgrounds, and are useful in measuring the false alarm rate of the system. Test Set 2 is a subset of the FERET database [16, 17]. Each image contains one face, and has (in most cases) a uniform background and good lighting. There are a wide variety of faces in the database, which are taken at a variety of angles. Thus these images are more useful for checking the angular sensitivity of the detector, and less useful for measuring the false alarm rate.

The outputs from our face detection networks are not binary. The neural network produce real values between 1 and -1, indicating whether or not the input contains a face. 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 presented in Fig. 10, 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.

Table 1 shows the performance of different versions of the detector on Test Set 1. The four columns show the number of faces missed (out of 507), the detection rate, the total number of false detections, and the false detection rate. The last rate is in terms of the number of 20x20 pixel windows that must be examined, which is approximately 3.3 times the number of pixels in an image (taking into account all the levels in the input pyramid). First we tested four networks working alone, then examined the effect of overlap elimination and collapsing multiple detections, and tested arbitration using ANDing, ORing, voting, and neural networks. 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 networks were based on Networks 1 and 2, while voting and network arbitration were based on Networks 1, 2, and 3. The neural network arbitrators were trained using the images from which the face examples were extracted. Three different architectures for the network arbitrator were used. The first used 5 hidden units, as shown in Fig. 8. The second used two hidden layers of 5 units each, with complete connections between each layer, and additional connections between the first hidden layer and the output. The last architecture was a simple perceptron, with no hidden units.

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 the table, 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 $449,184$ windows to down to 1 in $41,549,605$, 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 77.9%. On the other hand, System 12, which is based on ORing, has a higher detection rate of 90.3% 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.

Systems 14, 15, and 16, all of which use neural network-based arbitration among three networks, yield detection and false alarm rates between those of Systems 10 and 11. System 13, which uses voting among three networks, has an accuracy between that of Systems 11 and 12. System 17 will be described in the next section.

Table 2 shows the result of applying each of the systems to images in Test Set 2 (a subset of public portion of the FERET database [16, 17]). We partitioned the images into three groups, based on the nominal angle of the face with respect to the camera: frontal faces, faces at an angle 15° from the camera, and faces at an angle of 22.5°[4]. The direction of the face varies significantly within these groups. As can be seen from the table, the detection rate for systems arbitrating two networks ranges between 97.8% and 100.0% for frontal and 15° faces, while for 22.5° faces, the detection rate is between 91.5% and 97.4%. This difference is because the training set contains mostly frontal faces. It is interesting to note that the systems generally have a higher detection rate for faces at an angle of 15° than for frontal faces. The majority of people whose frontal faces are missed are wearing glasses which are reflecting light into the camera. The detector is not trained on such images, and expects the eyes to be darker than the rest of the face. Thus the detection rate for such faces is lower.

Based on the results shown in Tables 1 and 2, we concluded that both Systems 11 and 15 make acceptable tradeoffs between the number of false detections and the detection rate. Because System 11 is less complex than System 15 (using only two networks rather than a total of four), it is preferable. System 11 detects on average 86.2% of the faces, with an average of one false detection per $3,613,009$ 20x20 pixel windows examined in Test Set 1. Figs. 11, 12, and 13 show example output images from System 11 on images from Test Set 1[5].

4 Improving the Speed

In this section, we briefly discuss some methods to improve the speed of the system. The work described is preliminary, and is not intended to be an exhaustive exploration of methods to optimize the execution time.

The dominant factor in the running time of the system described thus far is the number of 20x20 pixel windows which the neural networks must process. Applying two networks to a 320x240 pixel image (246766 windows) on a 200 MHz R4400 SGI Indigo 2 takes approximately 383 seconds. The computational cost of the arbitration steps is negligible in comparison, taking less than one second to combine the results of the two networks over all positions in the image.

Recall that the amount of position invariance in the pattern recognition component of our system determines how many windows must be processed. In the related task of license plate detection, this was exploited to decrease the number of windows that must be processed [22]. The idea was to make the neural network be invariant to translations of about 25% of the size of the license plate. Instead of a single number indicating the existence of a face in the window, the output of Umezaki's network is an image with a peak indicating the location of the license plate. These outputs are accumulated over the entire image, and peaks are extracted to give candidate locations for license plates.

The same idea can be applied to face detection. The original detector was trained to detect a 20x20 face centered in a 20x20 window. We can make the detector more flexible by allowing the same 20x20 face to be off-center by up to 5 pixels in any direction. To make sure the network can still see the whole face, the window size is increased to 30x30 pixels. Thus the center of the face will fall within a 10x10 pixel region at the center of the window. As before, the network has a single output, indicating the presence or absence of a face. This detector can be moved in steps of 10 pixels across the image, and still detect all faces that might be present. The scanning method is illustrated in Fig. 14, which shows the input image pyramid and which of the 10x10 pixel regions are classified as containing the centers of faces. An architecture with an image output was also tried, which yielded about the same detection accuracy, but required more computation. The network was trained using the same bootstrap procedure as described earlier. The windows are preprocessed with histogram equalization before they are passed to the network.

As can be seen from the figure, this network has many more false detections than the detectors described earlier. To improve the accuracy, we treat each detection by the 30x30 detector as a candidate, and use the 20x20 detectors described earlier to verify it. Since the candidate faces are not precisely located, the verification network's 20x20 window must be scanned over the 10x10 pixel region potentially containing the center of the face. A simple arbitration strategy, ANDing, is used to combine the outputs of two verification networks. The heuristic that faces rarely overlap can also be used to reduce computation, by first scanning the image for large faces, and at smaller scales not processing locations which overlap with any detections found so far. The results of these verification steps are illustrated on the right side of Fig. 14.

With these modifications, the processing time for a typical 320x240 image is about 7.2 seconds on a 200 MHz R4400 SGI Indigo 2. To examine the effect of these changes on the accuracy of the system, it was applied to the test sets used in the previous section. The results are listed as System 17 in Tables 1 and 2. As can be seen, this system has detection and false alarm rates comparable with the most conservative of the other systems, System 10.

Further performance improvements can be made if one is analyzing many pictures taken by a stationary camera. By taking a picture of the background scene, one can determine which portions of the picture have changed in a newly acquired image, and analyze only those portions of the image. Similarly, a skin color detector like the one presented in [9] can restrict the search region. These techniques, taken together, have proven useful in building an almost real-time version of the system suitable for demonstration purposes, which can process a 320x240 image in 2 to 4 seconds, depending on the image complexity.

5 Comparison to Other Systems

Sung and Poggio developed a face detection system based on clustering techniques [21]. 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 "nonface" clusters. Two distance metrics measure the distance of an input image to the prototype clusters, the first measuring the distance between the test pattern and the cluster's 75 most significant eigenvectors, and the second measuring the Euclidean distance between the test pattern and its projection in the 75 dimensional subspace. 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. 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 3 shows the accuracy of their system on a set of 23 images (a portion of Test Set 1), along with the results of our system using the heuristics employed by Systems 10, 11, and 12 in Table 1. In [21], 149 faces were labelled in this test set, while we labelled 155. Some of these faces are difficult for either system to detect. Assuming that Sung and Poggio were unable to detect any of the six additional faces we labelled, the number of faces missed by their system is six more than listed in their paper. Table 3 shows that for equal numbers of false detections, we can achieve slightly higher detection rates.

Osuna, Freund, and Girosi [14] have recently investigated face detection using a framework similar to that used in [21] and in our own work. However, they use a "support vector machine" to classify images, rather than a clustering-based method or a neural network. The support vector machine has a number of interesting properties, including the fact that it makes the boundary between face and nonface images more explicit. The result of their system on the same 23 images used in [21] is given in Table 3; the accuracy is currently slightly poorer than the other two systems for this small test set.

As with Sung and Poggio's work, Moghoddam and Pentland's approach uses a two component distance measure, but combines the two distances in a principled way based on the assumption that the distribution of each cluster is Gaussian [13]. The clusters are used together as a multi-modal Gaussian distribution, giving a probability distribution for all face images. Faces are detected by measuring how well each window of the input image fits the distribution, and setting a threshold. This detection technique has been applied to faces, and to the detection of smaller features like the eyes, nose, and mouth.

Moghaddam and Pentland's system, along with several others, was tested in the FERET evaluation of face recognition methods [16,17]. Although the actual detection error rates are not reported, an upper bound can be derived from the recognition error rates. The recognition error rate, averaged over all the tested systems, for frontal photographs taken in the same sitting is less than 2% (see the rank 50 results in Figure 4 of [16]). This means that the number of images containing detection errors, either false alarms or missing faces, was less than 2% of all images. Anecdotally, the actual error rate is significantly less than 2%. As shown in Table 2, our system using the configuration of System 11 achieves a 2% error rate on frontal faces. Given the large differences in performance of our system on Test Set 1 and the FERET images, it is clear that these two test sets exercise different portions of the system. The FERET images examine the coverage of a broad range of face types under good lighting with uncluttered backgrounds, while Test Set 1 tests the robustness to variable lighting and cluttered backgrounds.

The candidate verification process used to speed up our system, described in Section 4, is similar to the detection technique presented in [23]. In that work, two networks were used. The first network has a single output, and like our system it is trained to produce a positive value for centered faces, and a negative value for nonfaces. 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. The network must be applied at every pixel position, but it runs quickly because of the its 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 the preprocessing to have a restricted form, such that it takes as input the entire image, and produces as output a new image. The nonlinear 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 nonfaces. All candidates which produce a positive response from the second network are output as detections. One possible problem with this work 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 training technique described here and in [21].

In recent work, Colmenarez and Huang presented a statistically based method for face detection [4]. Their system builds probabilistic models of the sets of faces and nonfaces, and compares how well each input window compares with these two categories. When applied to Test Set 1, their system achieves a detection rate between 86.8% and 98.0%, with between 6133 and 12758 false detections, respectively, depending on a threshold. These numbers should be compared to Systems 1 through 4 in Table 1, which have detection rates between 90.9% and 92.1%, with between 738 and 945 false detections. Although their false alarm rate is significantly higher, their system is quite fast. It would be interesting to use this system as a replacement for the candidate detector described in Section 4.

6 Conclusions and Future Research

Our algorithm can detect between 77.9% and 90.3% of faces in a set of 130 test 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. A fast version of the system can process a 320x240 pixel image in 2 to 4 seconds on a 200 MHz R4400 SGI Indigo 2.

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 each head orientation, and the results could be combined using arbitration methods similar to those presented here. Preliminary work in this area indicates that detecting profiles views of faces is more difficult than detecting frontal views, because they have fewer stable features and because the input window will contain more background pixels. We have also applied the same algorithm for the detection of car tires and human eyes, although more work is needed.

Even within the domain of detecting frontal views of faces, more work remains. When an image sequence is available, temporal coherence can focus attention on particular portions of the images. As a face moves about, its location in one frame is a strong predictor of its location in next frame. Standard tracking methods, as well as expectation-based methods [2], can be applied to focus the detector's attention. Other methods of improving system performance include obtaining more positive examples for training, or applying more sophisticated image preprocessing and normalization techniques.

One application of this work is in the area of media technology. Every year, improved technology provides cheaper and more efficient ways of storing and retrieving visual 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. Systems utilizing the detector described above allow a user to make requests of the form "Show me the people who appear in this video" [18, 20] or "Which images on the World Wide Web contain faces?" [6] and to have their queries answered automatically.

Acknowledgements

The authors would like to thank 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 for training, respectively. Michael Smith (at CMU) provided some digitized television images for testing purposes. Test Set 2 consists of facial images from the FERET database, collected under the ARPA/ARL FERET program [17]. We also thank Eugene Fink, Xue-Mei Wang, Hao-Chi Wong, Tim Rowley, Kaari Flagstad, and the reviewers for their comments on earlier versions of this paper.

This work was partially supported by a grant from Siemens Corporate Research, Incorporated, by the Department of the Army, Army Research Office under grant number DAAH04-94-G-0006, and by the Office of Naval Research under grant number N00014-95-1-0591. Shumeet Baluja received support from a National Science Foundation Graduate Fellowship, and 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 necessarily representing official policies or endorsements, either expressed or implied, of the sponsoring agencies.

References

[1] Shumeet Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[2] Shumeet Baluja. *Expectation-Based Selective Attention*. PhD thesis, Carnegie Mellon University Computer Science Department, October 1996. Available as CS Technical Report CMU-CS-96-182.

[3] Gilles Burel and Dominique Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15:963–967, October 1994.

[4] Antonio J. Colmenarez and Thomas S. Huang. Face detection with information-based maximum discrimination. In *Computer Vision and Pattern Recognition*, pages 782–787, 1997.

[5] Harris Drucker, Robert Schapire, and Patrice Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.

[6] Charles Frankel, Michael J. Swain, and Vassilis Athitsos. WebSeer: An image search engine for the world wide web. Technical Report TR-96-14, University of Chicago, August 1996.

[7] Venu Govindaraju. Locating human faces in photographs. *International Journal of Computer Vision*, 19(2):129–146, 1996.

[8] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.

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

[10] 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.

[11] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Fifth International Conference on Computer Vision*, pages 637–644, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.

[12] S. H. Lin, S. Y. Kung, and L. J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks, Special Issue on Artificial Neural Networks and Pattern Recognition*, 8(1), January 1997.

[13] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *Fifth International Conference on Computer Vision*, pages 786–793, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.

[14] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[15] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[16] P. Jonathon Phillips, Hyeonjoon Moon, Patrick Rauss, and Syed A. Rizvi. The FERET evaluation methodology for face-recognition algorithms. In *Computer Vision and Pattern Recognition*, pages 137–143, 1997.

[17] P. Jonathon Phillips, Patrick J. Rauss, and Sandor Z. Der. FERET (face recognition technology) recogition algorithm development and test results. Technical Report ARL-TR-995, Army Research Laboratory, October 1996.

[18] Shin'ichi Satoh and Takeo Kanade. Name-it: Association of face and name in video. In *Computer Vision and Pattern Recognition*, pages 368–373, 1997.

[19] Pawan Sinha. Object recognition via image invariants: A case study. *Investigative Opthalmology and Visual Science*, 35(4), march 1994.

[20] Michael A. Smith and Takeo Kanade. Video skimming and characterization through the combination of image and language understanding techniques. In *Computer Vision and Pattern Recognition*, pages 775–781, 1997.

[21] Kah-Kay Sung. *Learning and Example Selection for Object nad Pattern Detection*. PhD thesis, MIT AI Lab, January 1996. Available as AI Technical Report 1572.

[22] Tazio Umezaki. Personal communication, 1995.

[23] 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.

[24] 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.

[25] Gaungzheng Yang and Thomas S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[26] Kin Choong Yow and Roberto Cipolla. Feature-based human face detection. Technical Report CUED/F-INFENG/TR 249, Department of Engineering, University of Cambridge, England, 1996.

Biographies

Henry A. Rowley is currently a Ph.D. student at Carnegie Mellon University in the Computer Science Department. His research work involves using machine learning techniques improve the performance of object detection and recognition algorithms in computer vision. He received B.S. degrees in Computer Science and Electrical Engineering from the University of Minnesota in 1992.

Shumeet Baluja received his B.S. degree in computer science from the University of Virginia, Charlottesville, Virginia, in 1992, and completed his Ph.D. in computer science from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1996. He is currently a research scientist at Justsystem Pittsburgh Research Center, and an adjunct faculty member in the Computer Science Department and The Robotics Institute at Carnegie Mellon University.

His research interests include the integration of machine learning and computer vision, mechanisms for selective attention in vision, artificial neural networks and their applications, and high-dimensional optimization.

Takeo Kanade received a Doctoral degree in electrical engineering from Kyoto University, Japan, in 1974.

After holding a faculty position at the Department of Information Science, Kyoto University, he joined Carnegie Mellon University in 1980, where he is currently the U. A. Helen Whitaker Professor of Computer Science and Director of the Robotics Institute.

Dr. Kanade has made technical contributions in multiple areas of robotics: vision, manipulators, autonomous mobile robots, and sensors. He has written more than 150 technical papers and reports in these areas. He has been the principle investigator of several major vision and robotics projects at Carnegie Mellon. In the area of education, he was a founding chairperson of Carnegie Mellon's Robotics Ph.D. program, probably the first of its kind. Dr. Kanade is a Founding Fellow of the American Association of Artificial Intelligence, and the founding editor of the *International Journal of Computer Vision*. He has served for many government, industry, and university advisory or consultant committees, including Aeronautics and Space Engineering Board (ASEB) of the National Research Council, NASA's Advanced Technology Advisory Committee (a congressionally mandated committee) and the Advisory Board of the Canadian Institute for Advanced Research.

Notes

[*]Henry A. Rowley and Takeo Kanade are with the Department of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, 15213, USA. Shumeet Baluja is with the Justsystem Pittsburgh Research Center, 4616 Henry Street, Pittsburgh, PA, 15213, USA, and is also associated with the Department of Computer Science and the Robotics Institute at Carnegie Mellon University. E-mail: `har@cs.cmu.edu, tk@cs.cmu.edu, baluja@jprc.com`.

[1]An interactive demonstration of the system 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 to see the detection results for pictures submitted by other people.

[2]Dr. Woodward Yang at Harvard provided over 400 mug-shot images which are part of the training set.

[3]These images are available over the World Wide Web, at the URL `http://www.cs.cmu.edu/~har/faces.html`.

[4]Specifically, we used images from groups 1 and 3, with labels `fa` and `fb` for the frontal group, `rb` and `rc` for the $15°$ group, and `ql` and `qr` for the $22.5°$ group.

[5]After painstakingly trying to arrange these images compactly by hand, we decided to use a more systematic approach. These images were laid out automatically by the PBIL optimization algorithm [1]. The objective function tries to pack images as closely as possible, by maximizing the amount of space left over at the bottom of each page.

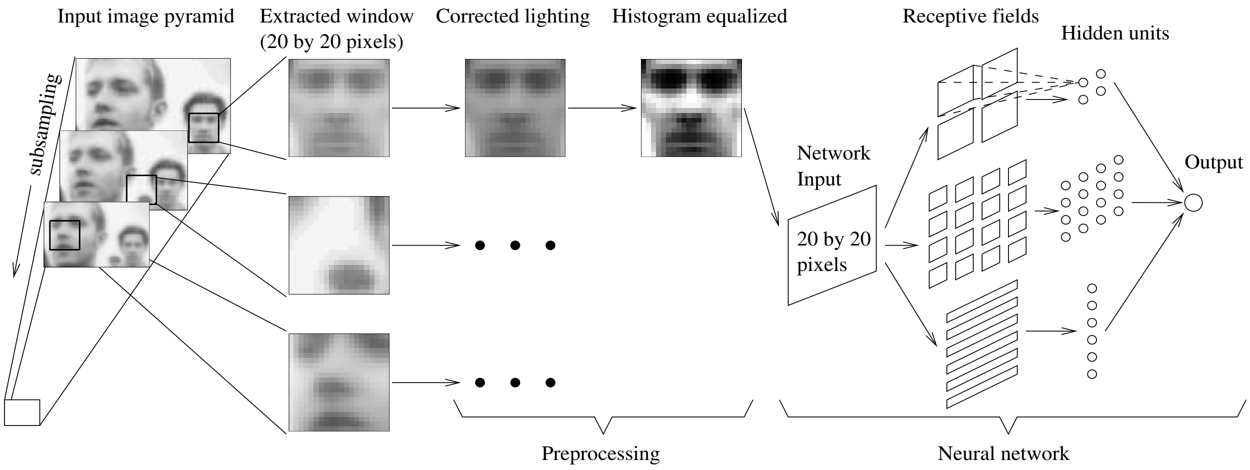

Figure 1: The basic algorithm used for face detection.

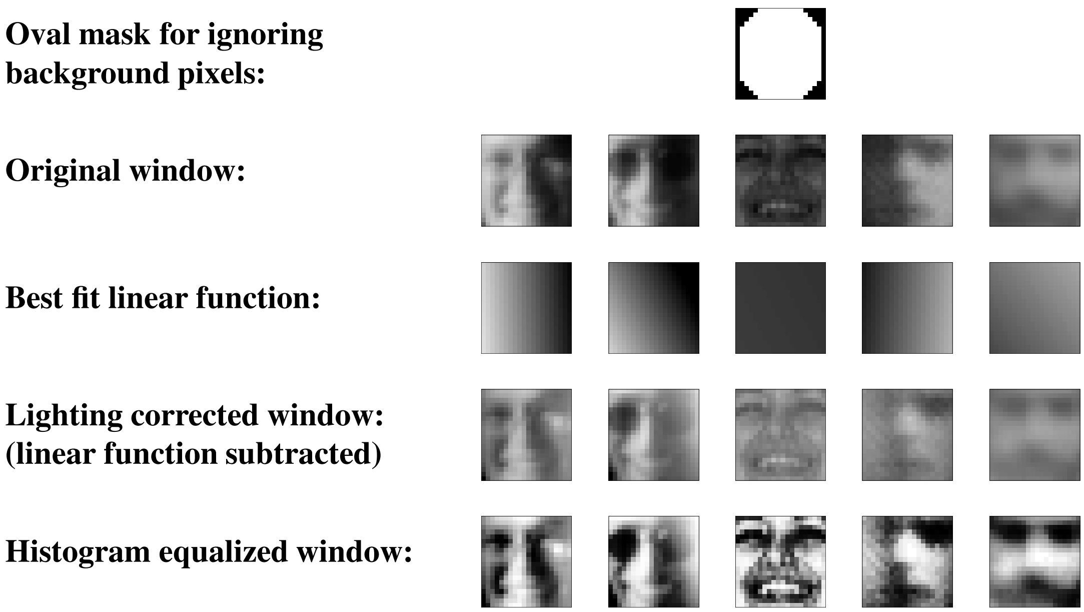

Figure 2: The steps in preprocessing a window. First, a linear function is fit to the intensity values in the window, and then subtracted out, correcting for some extreme lighting conditions. Then, histogram equalization is applied, to correct for different camera gains and to improve contrast. For each of these steps, the mapping is computed based on pixels inside the oval mask, and then applied to the entire window.

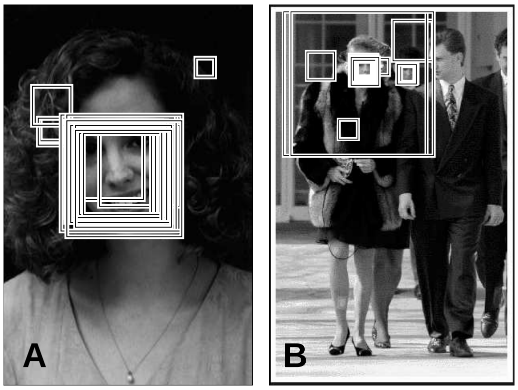

Figure 3: Images with all the above threshold detections indicated by boxes.

Figure 4: Example face images (the authors), randomly mirrored, rotated, translated, and scaled by small amounts.

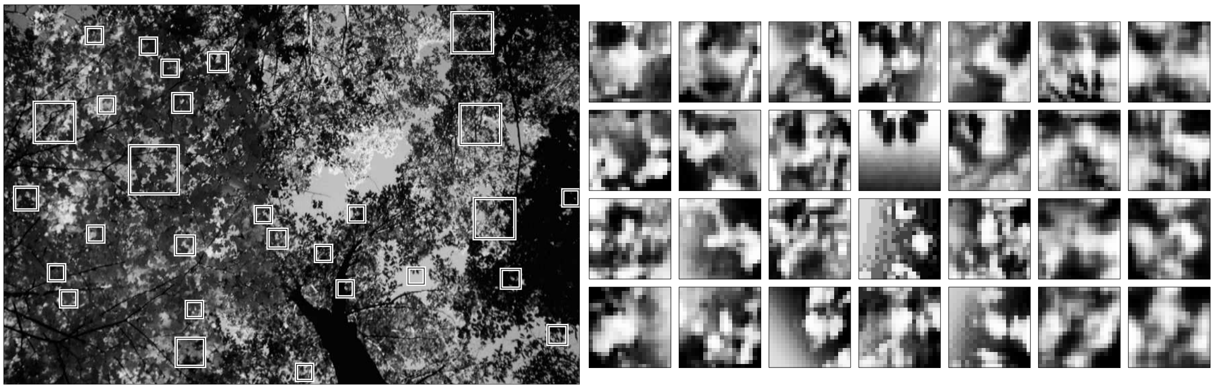

Figure 5: During training, the partially-trained system is applied to images of scenery which do not contain faces (like the one on the left). Any regions in the image detected as faces (which are expanded and shown on the right) are errors, which can be added into the set of negative training examples.

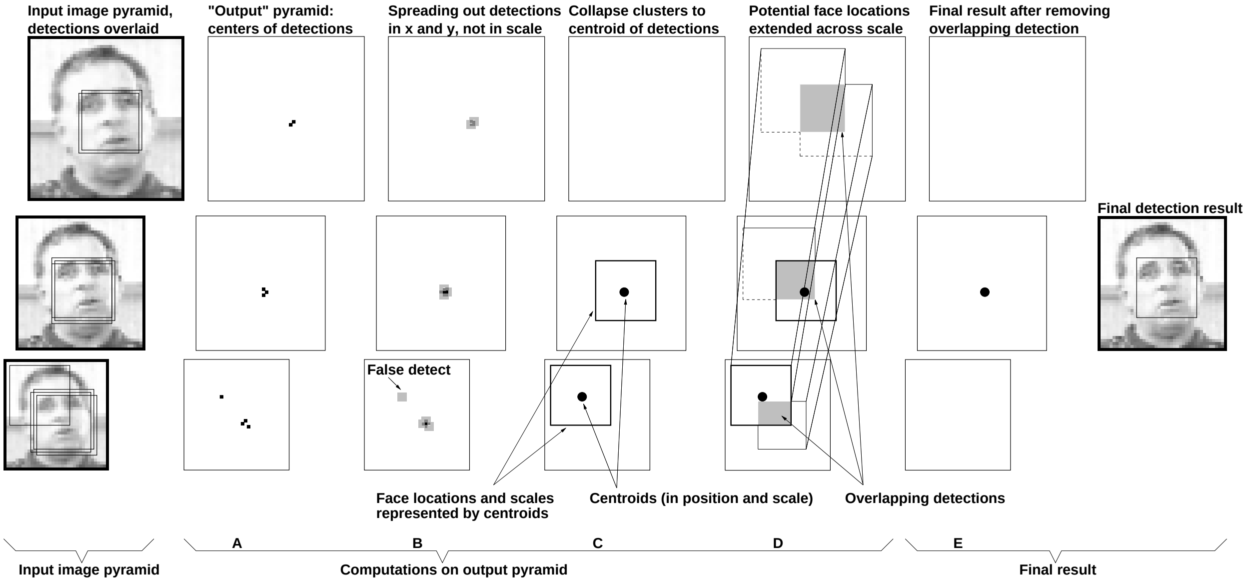

Figure 6: The framework for merging multiple detections from a single network: A) The detections are recorded in an "output" pyramid. B) The detections are "spread out" and a threshold is applied. C) The centroids in scale and position are computed, and the regions contributing to each centroid are collapsed to single points. In the example shown, this leaves only two detections in the output pyramid. D) The final step is to check the proposed face locations for overlaps, and E) to remove overlapping detections if they exist. In this example, removing the overlapping detection eliminates what would otherwise be a false positive.

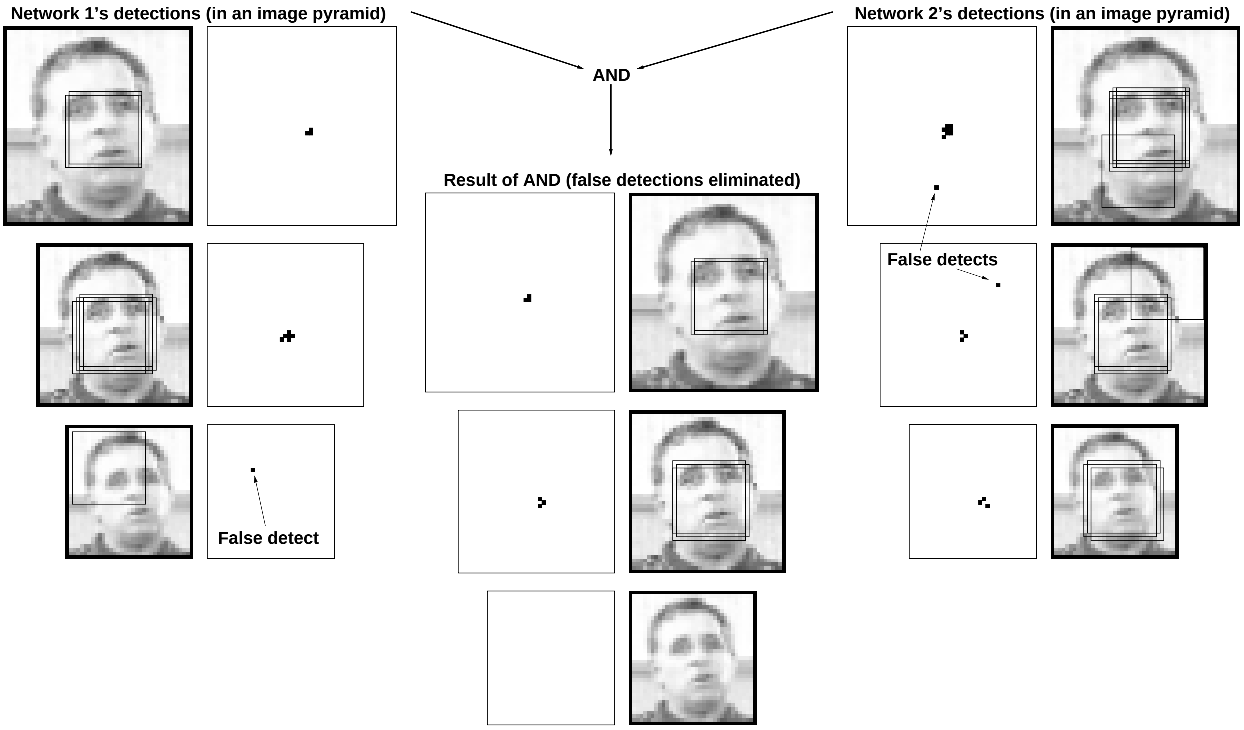

Figure 7: ANDing together the outputs from two networks over different positions and scales can improve detection accuracy.

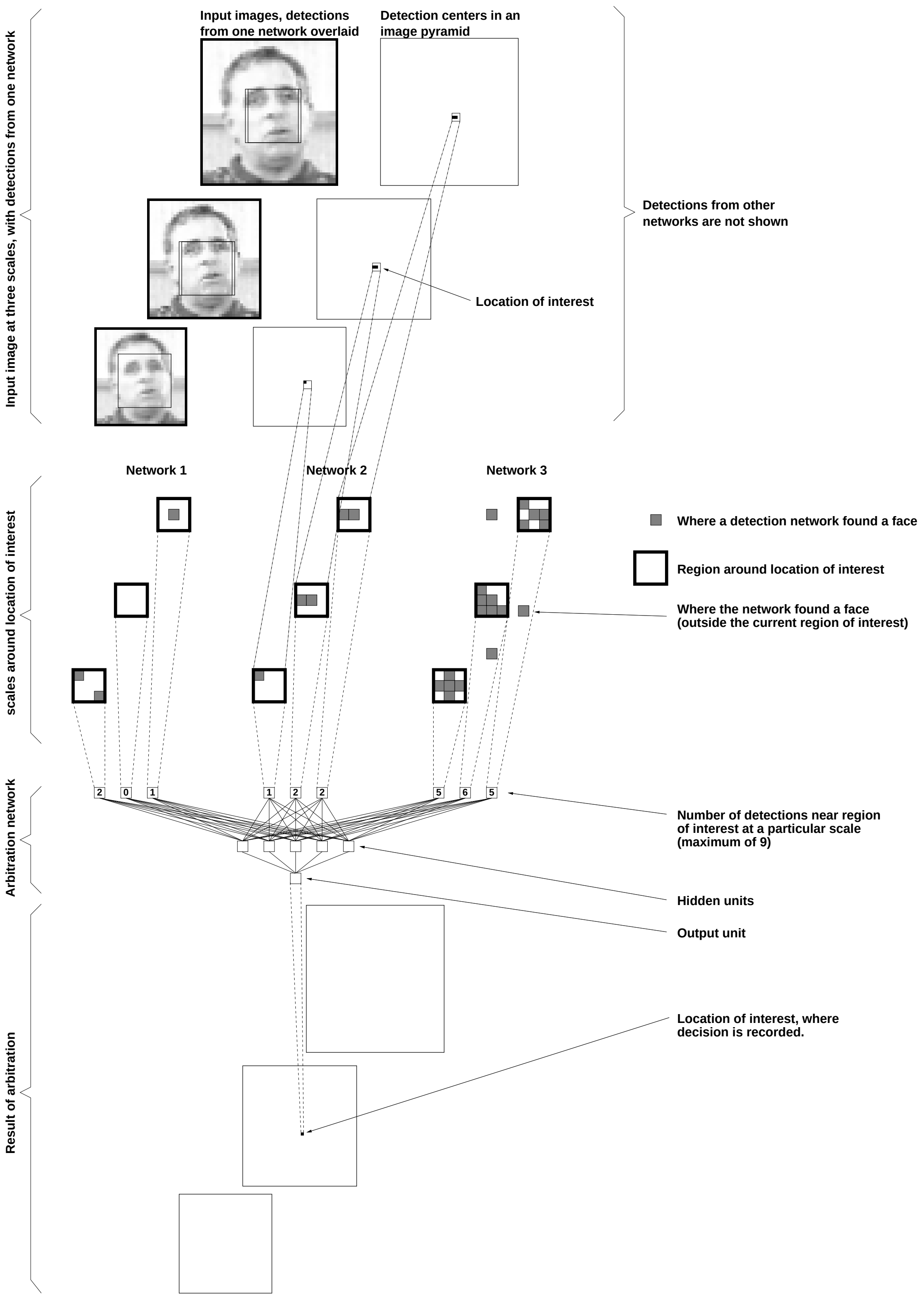

Figure 8: The inputs and architecture of the arbitration network which arbitrates among multiple face detection networks.

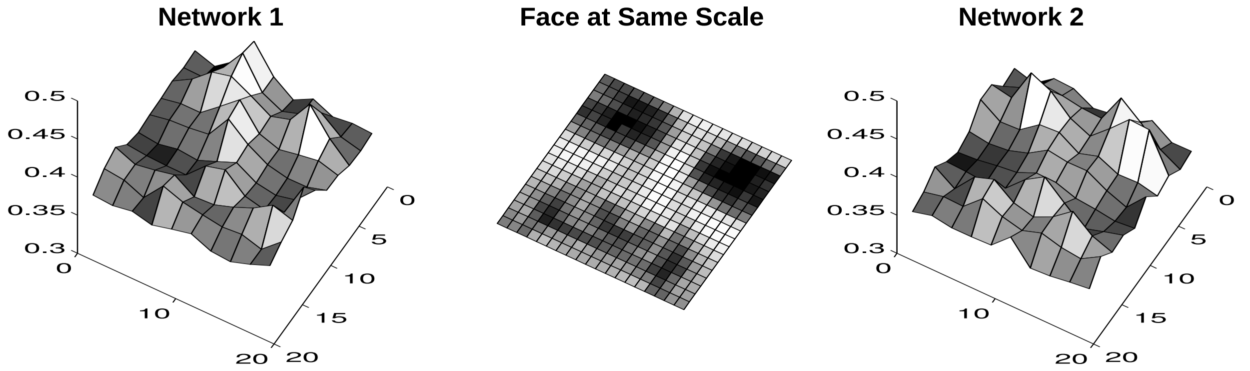

Figure 9: Error rates (vertical axis) on a test created by adding noise to various portions of the input image (horizontal plane), for two networks. Network 1 has two copies of the hidden units shown in Fig. 1 (a total of 58 hidden units and 2905 connections), while Network 2 has three copies (a total of 78 hidden units and 4357 connections).

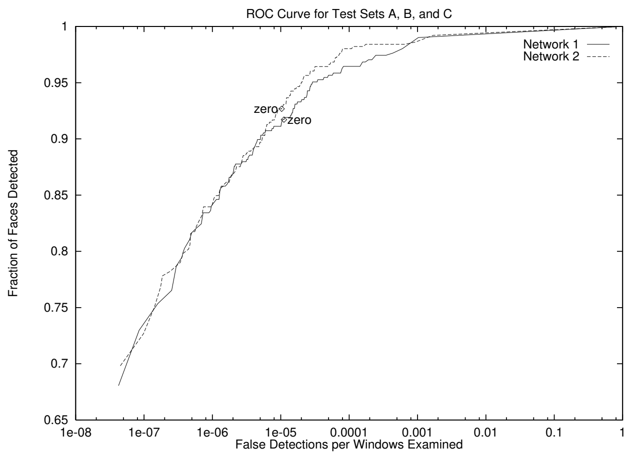

Figure 10: The detection rate plotted against false positive rates as the detection threshold is varied from -1 to 1, for the same networks as Fig. 9. The performance was measured over all images from Test Set 1. The points labelled "zero" are the zero threshold points which are used for all other experiments.

Table 1: Detection and error rates for Test Set 1, which consists of 130 images and contains 507 frontal faces. It requires the system to examine a total of 83099211 20x20 pixel windows.

Type	System	Missed faces	Detect rate	False detects	False detect rate
Single network, no heuristics	1) Network 1 (2 copies of hidden units (52 total), 2905 connections)	45	91.1%	945	1/87935
	2) Network 2 (3 copies of hidden units (78 total), 4357 connections)	38	92.5%	862	1/96402
	3) Network 3 (2 copies of hidden units (52 total), 2905 connections)	46	90.9%	738	1/112600
	4) Network 4 (3 copies of hidden units (78 total), 4357 connections)	40	92.1%	819	1/101464
Single network, with heuristics	5) Network 1 → threshold(2,1) → overlap elimination	48	90.5%	570	1/145788
	6) Network 2 → threshold(2,1) → overlap elimination	42	91.7%	506	1/164227
	7) Network 3 → threshold(2,1) → overlap elimination	49	90.3%	440	1/188861
	8) Network 4 → threshold(2,1) → overlap elimination	42	91.7%	484	1/171692
Arbitrating among two networks	9) Networks 1 and 2 → AND(0)	68	86.6%	79	1/1051888
	10) Networks 1 and 2 → AND(0) → threshold(2,3) → overlap elimination	112	77.9%	2	1/41549605
	11) Networks 1 and 2 → threshold(2,2) → overlap elimination → AND(2)	70	86.2%	23	1/3613009
	12) Networks 1 and 2 → thresh(2,2) → overlap elim → OR(2) → thresh(2,1) → overlap elimination	49	90.3%	185	1/449184
Arbitrating among three networks	13) Networks 1, 2, 3 → voting(0) → overlap elimination	59	88.4%	99	1/839385
	14) Networks 1, 2, 3 → network arbitration (5 hidden units) → thresh(2,1) → overlap elimination	79	84.4%	16	1/5193700
	15) Networks 1, 2, 3 → network arbitration (10 hidden units) → thresh(2,1) → overlap elimination	83	83.6%	10	1/8309921
	16) Networks 1, 2, 3 → network arbitration (perceptron) → thresh(2,1) → overlap elimination	84	83.4%	12	1/6924934
Fast version	17) Candidate verification method described in Section 4	117	76.9%	8	1/10387401

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 occurring 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.

Table 2: Detection and error rates for Test Set 2 (the FERET database)

Type	System	Frontal Faces		15° Angle		22.5° Angle	
	Number of Images		1001		241		378
	Number of Faces		1001		241		378
	Number of Windows		255129875		61424875		96342750
		# miss /	Detect rate	# miss /	Detect rate	# miss /	Detect rate
Type	System	False detects /	Rate	False detects /	Rate	False detects /	Rate
Single network, no heuristics	1) Net 1 (2 copies of hidden	5	99.5%	1	99.6%	7	98.1%
	units, 2905 connections)	1747	1/146038	447	1/137415	819	1/117634
	2) Net 2 (3 copies of hidden	5	99.5%	0	100.0%	11	97.1%
	units, 4357 connections)	1457	1/175106	481	1/127702	592	1/162741
	3) Net 3 (2 copies of hidden	4	99.6%	1	99.6%	8	97.9%
	units, 2905 connections)	1242	1/205418	374	1/164237	605	1/159244
	4) Net 4 (3 copies of hidden	5	99.5%	1	99.6%	15	96.0%
	units, 4357 connections)	1665	1/153231	458	1/134115	709	1/135885
Single network, with heuristics	5) Network 1 → threshold(2,1)	5	99.5%	1	99.6%	12	96.8%
	→ overlap elimination	643	1/396780	136	1/451653	263	1/366322
	6) Network 2 → threshold(2,1)	5	99.5%	0	100.0%	12	96.8%
	→ overlap elimination	458	1/557052	118	1/520549	146	1/659881
	7) Network 3 → threshold(2,1)	5	99.5%	1	99.6%	10	97.4%
	→ overlap elimination	421	1/606009	85	1/722645	133	1/724381
	8) Network 4 → threshold(2,1)	8	99.2%	1	99.6%	20	94.7%
	→ overlap elimination	563	1/453161	120	1/511873	207	1/465423
Arbitrating among two networks	9) Nets 1 and 2 → AND(0)	13	98.7%	1	99.6%	20	94.7%
		141	1/1809431	46	1/1335323	85	1/1133444
	10) Nets 1 and 2 → AND(0) →	22	97.8%	1	99.6%	32	91.5%
	threshold(2,3) → overlap elim	0	0/255129875	0	0/61424875	1	1/96342750
	11) Nets 1 and 2 → thresh(2,2)	8	99.2%	1	99.6%	17	95.5%
	→ overlap elim → AND(2)	12	1/21260822	3	1/20474958	3	1/32114250
	12) Nets1,2→thresh(2,2)→over	3	99.7%	0	100.0%	10	97.4%
	→OR(2)→thresh(2,1)→over	137	1/1862261	35	1/1754996	53	1/1817787
Arbitrating among three networks	13) Nets 1,2,3 → voting(0) →	31	96.9%	9	96.3%	28	92.6%
	overlap elimination	74	1/3447701	23	1/2670646	38	1/2535335
	14) Nets 1,2,3 → net arb (5	11	98.9%	1	99.6%	21	94.4%
	hiddens) → thresh(2,1) → over	5	1/51025975	1	1/61424875	3	1/32114250
	15) Nets 1,2,3 → net arb (10	13	98.7%	1	99.6%	21	94.4%
	hiddens) → thresh(2,1) → over	4	1/63782468	1	1/61424875	2	1/48171375
	16) Nets 1,2,3 → net arb	13	98.7%	1	99.6%	21	94.4%
	(percep) → thresh(2,1) → over	3	1/85043291	1	1/61424875	2	1/48171375
Fast version	17) Candidate verification	20	98.0%	2	99.2%	23	93.9%
	method described in Section 4	2	1/127564937	0	0/61424875	2	1/48171375

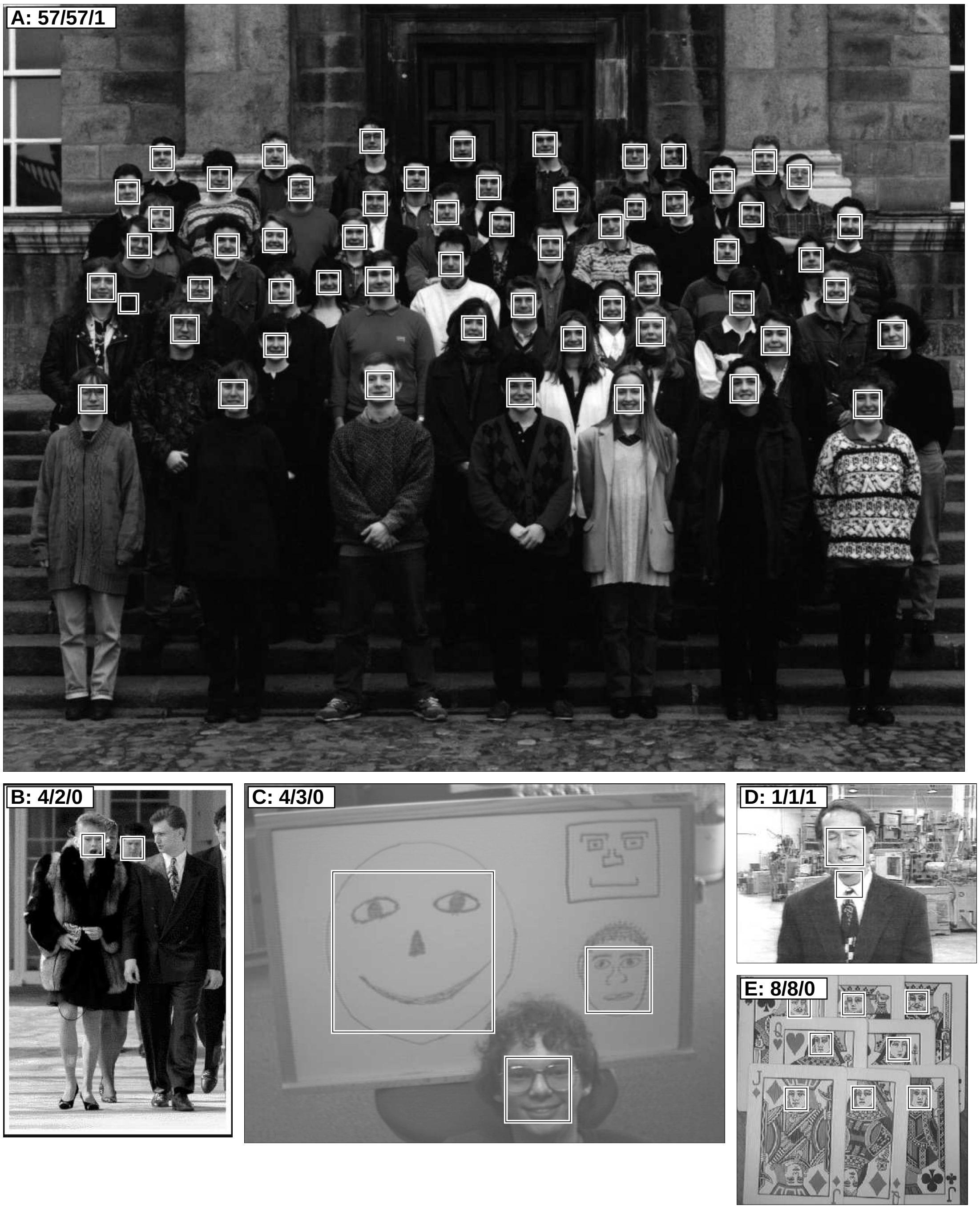

Figure 11: Output obtained from System 11 in Table 1 on images from Test Set 1. For each image, three numbers are shown: the number of faces in the image, the number of faces detected correctly, and the number of false detections. Some notes on specific images: Faces are missed in B (one due to occlusion, one due to large angle) and C (the stylized drawing was not detected at the same locations and scales by the two networks, and so is lost in the AND). False detections are present in A and D. Although the system was trained only on real faces, some hand drawn faces are detected in C and E. A was obtained from the World Wide Web, B and E were provided by Sung and Poggio at MIT, C is a CCD image, and D is a digitized television image.

Figure 12: Output obtained in the same manner as the examples in Fig. 11. Some notes on specific images: Faces are missed in C (one due to occlusion, one due to large angle), H (reflections off of glasses made the eyes appear brighter than the rest of the face), and K (due to large angle). False detections are present in B and K. Although the system was trained only on real faces, hand drawn faces are detected in B. A, B, J, K, and L were provided by Sung and Poggio at MIT, C, D, E, G, H, and M were scanned from photographs, F and I are digitized television images, and N is a CCD image.

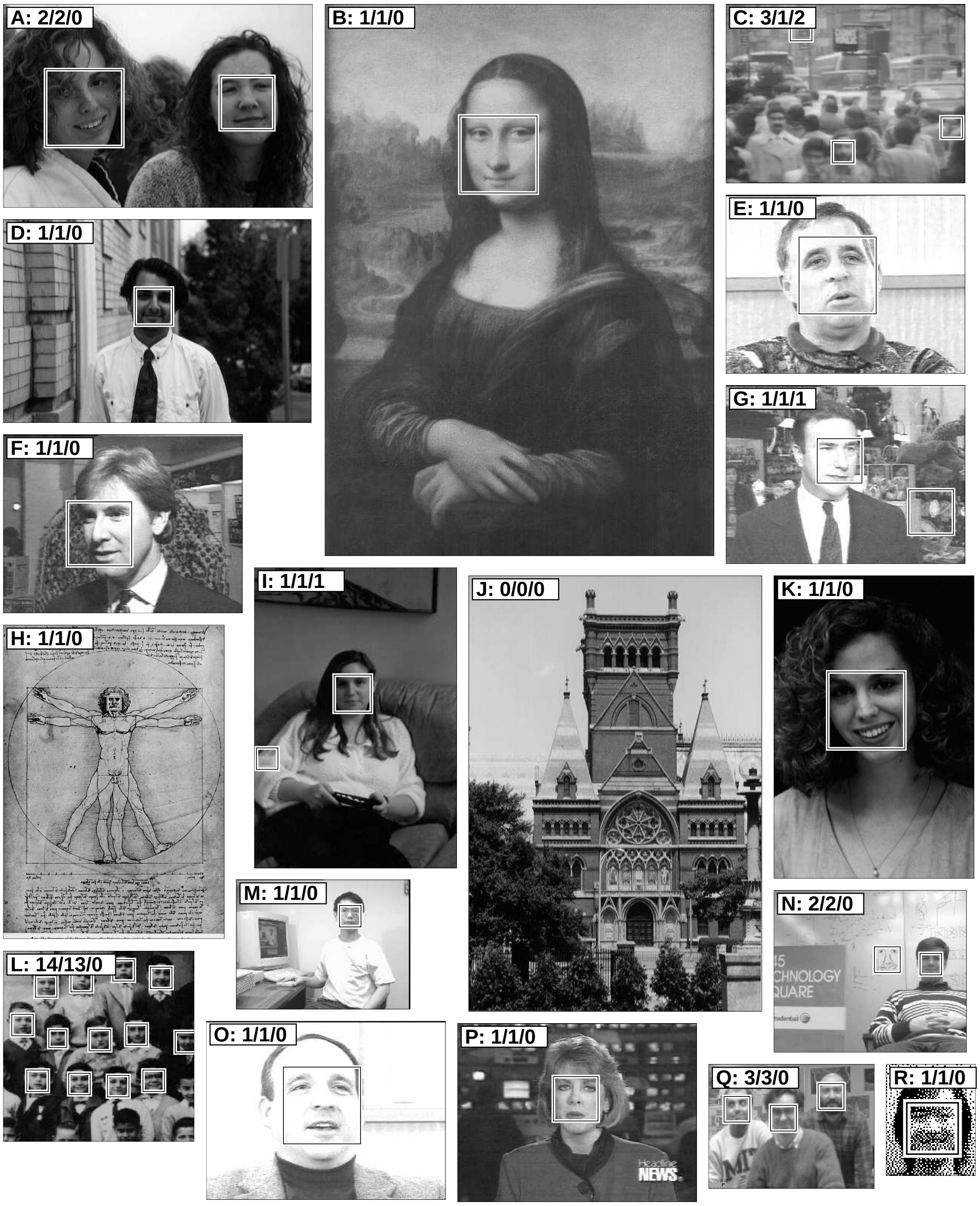

Figure 13: Output obtained in the same manner as the examples in Fig. 11. Some notes on specific images: Faces are missed in C (due to blurriness) and L (due to partial occlusion of the chin). False detections are present in C, G, and I. Although the system was trained only on real faces, hand drawn faces are detected in H and N. A, D, I, J, and K were scanned from photographs, B, H, and L were obtained from the World Wide Web, C, E, F, G, O, and P are digitized television images. M, N, and Q were provided by Sung and Poggio at MIT, and R is a dithered CCD image.

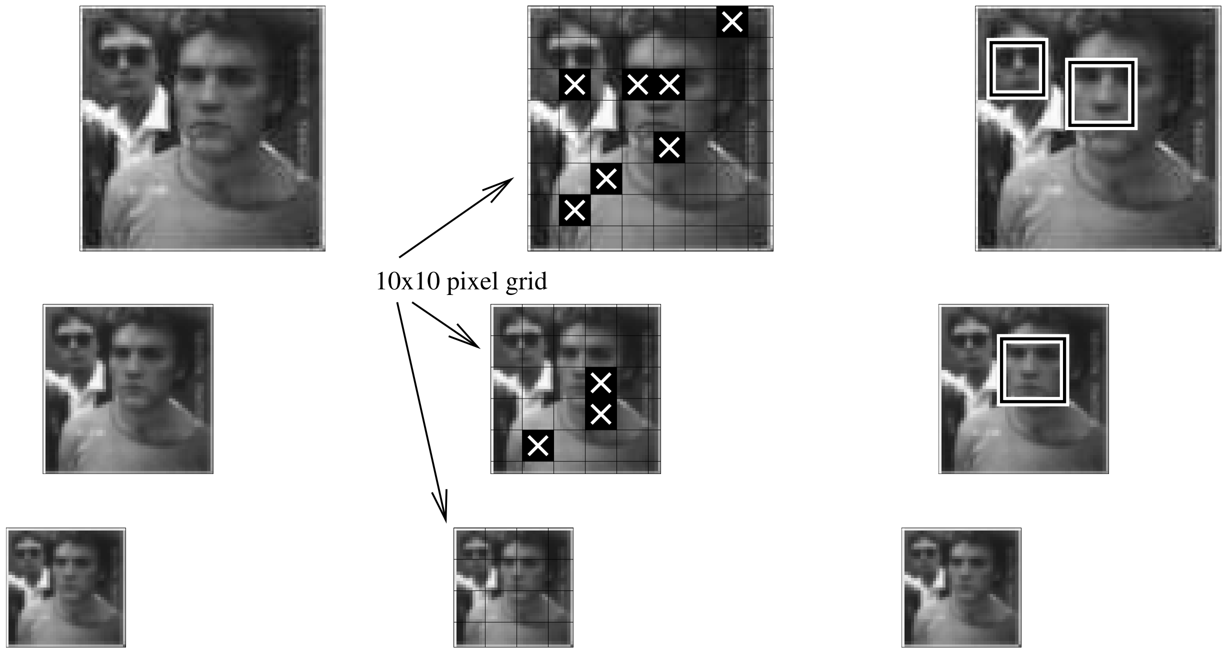

Figure 14: Illustration of the steps in the fast version of the face detector. On the left is the input image pyramid, which is scanned with a 30x30 detector that moves in steps of 10 pixels. The center of the figure shows the 10x10 pixel regions (at the center of the 30x30 detection windows) which the 20x20 detector believes contain the center of a face. These candidates are then verified by the detectors described earlier in the paper, and the final results are shown on the right.

Table 3: Comparison of the detectors in [21] and [14] and our system on a 23 image subset of Test Set 1, containing 155 faces.

System	Missed faces	Detect rate	False detects
10) Nets 1,2 → AND(0) → threshold(2,3) → overlap elimination	39	74.8%	0
11) Nets 1,2 → threshold(2,2) → overlap elimination → AND(2)	24	84.5%	8
12) Nets 1,2 → threshold(2,2) → overlap → OR(2) → threshold(2,1) → overlap	15	90.3%	42
Detector using a multi-layer network [21]	36	76.8%	5
Detector using perceptron [21]	28	81.9%	13
Support Vector Machine [14]	39	74.2%	20

Contact Information

Henry A. Rowley

```
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA   15213
Phone:  (412) 268-8734
Fax:    (412) 268-5576
E-mail: har@cs.cmu.edu
```

Shumeet Baluja

```
Justsystem Pittsburgh Research Center
4616 Henry Street
Pittsburgh, PA   15213
Phone:  (412) 683-8592
Fax:    (412) 683-4175
E-mail: baluja@jprc.com
```

Takeo Kanade

```
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA   15213
Phone:  (412) 268-3016
Fax:    (412) 268-5570
E-mail: tk@cs.cmu.edu
```