Journal of Machine Learning Research 8 (2007) 1197-1215 Submitted 8/05; Revised 2/07; Published 5/07

Synergistic Face Detection and Pose Estimation with Energy-Based Models

Margarita Osadchy RITA@CS.HAIFA.AC.IL
Department of Computer Science
University of Haifa
Mount Carmel, Haifa 31905,Israel

Yann Le Cun YANN@CS.NYU.EDU
The Courant Institute
New York University
New York, NY 10003, USA

Matthew L. Miller MLM@NEC-LABS.COM
NEC Labs America
Princeton NJ 08540, USA

Editor: Pietro Perona

Abstract

We describe a novel method for simultaneously detecting faces and estimating their pose in real time. The method employs a convolutional network to map images of faces to points on a low-dimensional manifold parametrized by pose, and images of non-faces to points far away from that manifold. Given an image, detecting a face and estimating its pose is viewed as minimizing an energy function with respect to the face/non-face binary variable and the continuous pose parameters. The system is trained to minimize a loss function that drives correct combinations of labels and pose to be associated with lower energy values than incorrect ones.

The system is designed to handle very large range of poses without retraining. The performance of the system was tested on three standard data sets—for frontal views, rotated faces, and profiles—is comparable to previous systems that are designed to handle a single one of these data sets.

We show that a system trained simuiltaneously for detection and pose estimation is more accurate on *both tasks* than similar systems trained for each task separately.[1]

Keywords: face detection, pose estimation, convolutional networks, energy based models, object recognition

1. Introduction

The detection of human faces in natural images and videos is a key component in a wide variety of applications of human-computer interaction, search and indexing, security, and surveillance. Many real-world applications would profit from *view-independent* detectors that can detect faces under a wide range of poses: looking left or right (yaw axis), up or down (pitch axis), or tilting left or right (roll axis).

In this paper we describe a novel method that can not only detect faces independently of their poses, but also simultaneously estimate those poses. The system is highly reliable, runs in real time

1. A more preliminary version of this work appears as: Osadchy et al. (2005).

©2007 Margarita Osadchy, Yan Le Cun and Matthew L. Miller.

on standard hardware, and is robust to variations in yaw ($\pm 90^\circ$), roll ($\pm 45^\circ$), pitch ($\pm 60^\circ$), as well as partial occlusions.

The method is motivated by the idea that multi-view face detection and pose estimation are so closely related that they should not be performed separately. The tasks are related in the sense that they could use similar features and internal representations, and must be robust against the same sorts of variation: skin color, glasses, facial hair, lighting, scale, expressions, etc. We suspect that, when trained together, each task can serve as an inductive bias for the other, yielding better generalization or requiring fewer training examples (Caruana, 1997).

To exploit the synergy between these two tasks, we train a learning machine to map input images to points in a low-dimensional space. In the low-dimensional output space we embed a *face manifold* which is parameterized by facial pose parameters (e.g., pitch, yaw, and roll). A convolutional network is trained to map face images to points on the face manifold that correspond to the pose of the faces and non-face images to points far away from that manifold. After training, a detection is performed by measuring whether the distance of the output point from the manifold is lower than a threshold. If the point is close to the manifold, indicating that a face is present in the image, its pose parameters can be inferred from the position of the projection of the point onto the manifold.

To map input images to points in the low-dimensional space, we employ a convolutional network architecture (LeCun et al., 1998). Convolutional networks are specifically designed to learn invariant representation of images. They can easily learn the type of shift-invariant local features that are relevant to face detection and pose estimation. More importantly, they can be replicated over large images (applied to every sub-windows in a large image) at a small fraction of the cost of applying more traditional classifiers to every sub-windows in an image. This is a *considerable advantage for building real-time systems*.

As a learning machine we use the recently proposed *Energy-Based Models* (EBM) that provide a description and the inference process and the learning process in a single, well-principled framework (LeCun and Huang, 2005; LeCun et al., 2006).

Given an input (an image), an Energy-Based Model associates an energy to each configuration of the variables to be modeled (the face/non-face label and the pose parameters in our case). Making an inference with an EBM consists in searching for a configuration of the variables to be predicted that minimizes the energy, or comparing the energies of a small number of configurations of those variables. EBMs have a number of advantages over probabilistic models: (1) There is no need to compute partition functions (normalization constants) that may be intractable; (2) because there is no requirement for normalization, the repertoire of possible model architectures that can be used is considerably richer. In our application we define an Energy-Based Model as a scalar-valued energy function of three variables: image, label, and pose, and we treat pose as a deterministic latent variable. Thus both label of an image and pose are inferred through the energy-minimization process.

Training an EBM consists in finding values of the trainable parameters (which parameterize the energy function) that associate low energies to "desired" configurations of variables, and high energies to "undesired" configurations. With probabilistic models, making the probability of some values large automatically makes the probabilities of other values small because of the normalization. With EBM's making the energy of desired configurations low may not necessarily make the energies of other configurations high. Therefore, one must be very careful when designing loss functions for EBMs. In our application to face detection we derive a new type of *contrastive loss function* that is tailored to such detection tasks.

The paper is organized as follows. First, some of the relevant prior works on multi-view face detection are briefly discussed. Section 2 discusses the synergy between pose estimation and face detection, and describes the basic methods for integrating them. Section 3 discusses the learning machine, and Section 4 gives the results of experiments conducted with our system. Section 5 draws some conclusions.

1.1 Previous Work

Learning-based approaches to face detection abound, including real-time methods (Viola and Jones, 2001), and approaches based on convolutional networks (Vaillant et al., 1994; Garcia and Delakis, 2002). Most multi-view systems take a *view-based* approach, which involves building separate detectors for different views and either applying them in parallel (Pentland et al., 1994; Sung and Poggio, 1998; Schneiderman and Kanade, 2000; Li et al., 2002) or using a pose estimator to select the most appropriate detector (Jones and Viola, 2003; Huang et al., 2004). Another approach is to estimate and correct in-plane rotations before applying a single pose-specific detector (Rowley et al., 1998b). Some attempts have been done in integrating pose search and detection, but in much smaller space of pose parameters (Fleuret and Geman, 2001).

Closer to our approach is that of Li et al. (2000), in which a number of Support Vector Regressors are trained to approximate smooth functions, each of which has a maximum for a face at a particular pose. Another machine is trained to convert the resulting values to estimates of poses, and a third machine is trained to convert the values into a face/non-face score. The resulting system is rather slow. See Yang et al. (2002) for survey of face detection methods.

2. Integrating Face Detection and Pose Estimation

To exploit the posited synergy between face detection and pose estimation, we must design a system that integrates the solutions to the two problems. Merely cascading two systems where the answer to one problem is used to assist in solving the other will not optimally take advantage of the synergy. Therefore, both answers must be derived from one underlying analysis of the input, and both tasks must be trained together.

Our approach is to build a trainable system that can map raw images X to points in a low-dimensional space (Figure 1). In that space, we pre-define a *face manifold* $F(Z)$ that we parameterize by the pose Z. We train the system to map face images with known poses to the corresponding points on the manifold. We also train it to map images of non-faces to points far away from the manifold. During recognition, the system maps the input image X to a point in the low dimensional space $G(X)$. The proximity of $G(X)$ to the manifold then tells us whether or not an image X is a face. By finding the pose parameters Z that correspond to the point on the manifold that is closest to the point $G(X)$ (projection), we obtain an estimate of the pose (Figure 2).

2.1 Parameterizing the Face Manifold

We will now describe the details of the parameterizations of the face manifold. Three criteria directed the design of the face manifold: (1) preserving the topology and geometry of the problem; (2) providing enough space for mapping the background images far from the manifold (since the proximity to the manifold indicates whether the input image contains a face); and (3) minimizing

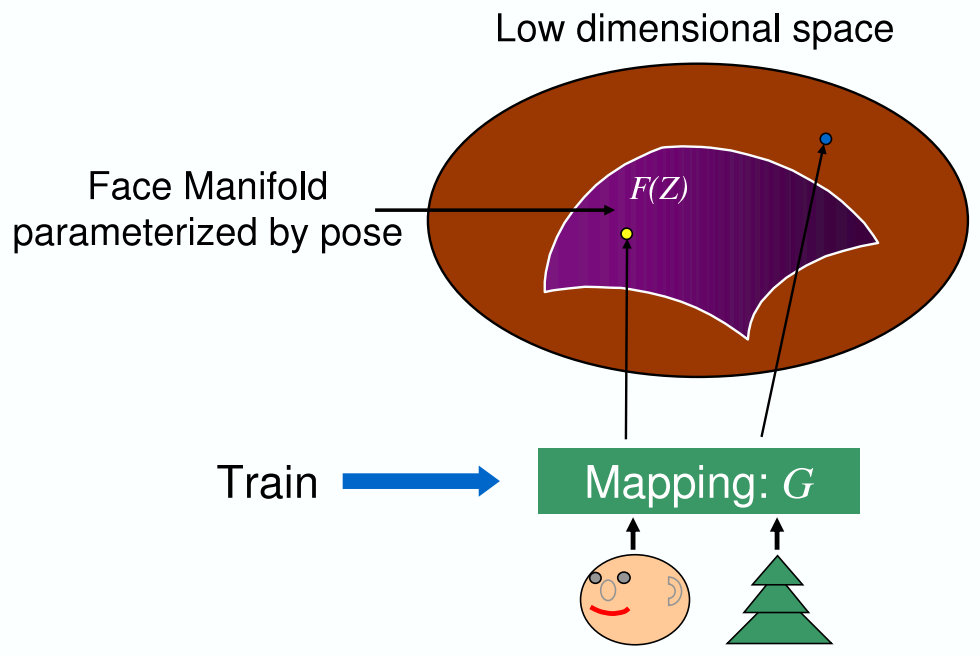

Figure 1: Manifold Mapping—Training

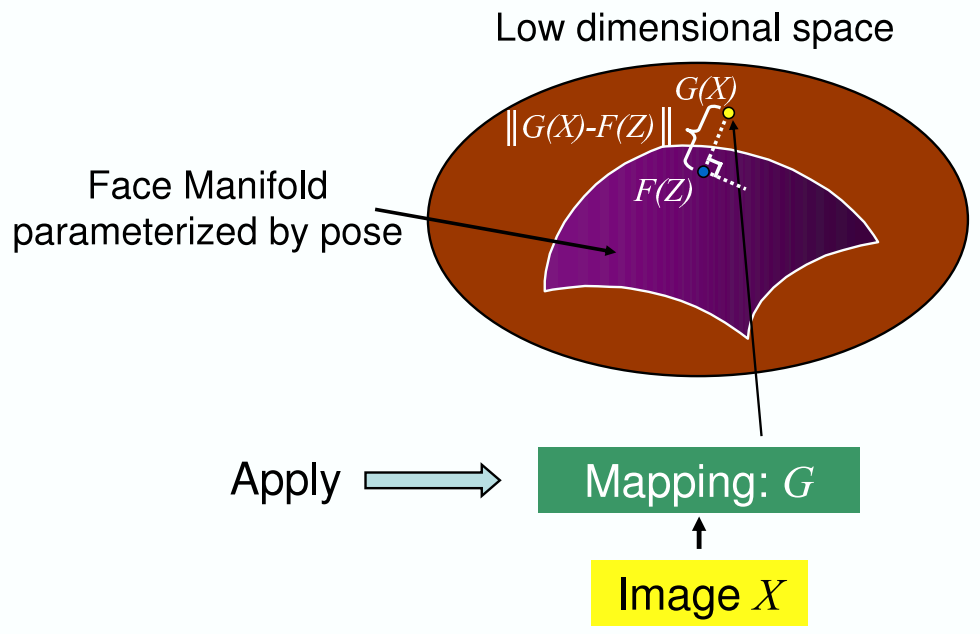

Figure 2: Manifold Mapping— Recognition and Pose Estimation.

the computational cost of finding the parameters of the closest point on the manifold to any point in the space.

Let's start with the simplest case of one pose parameter $Z = \theta$, representing, say, yaw. If we want to preserve the natural topology and geometry of the problem (the first criterion), the face manifold under yaw variations in the interval $[-90°, 90°]$ should be a half circle (with constant curvature). The natural way of representing a circle is with sine and cosine functions. In this case we embed the angle parameter in two dimensional space. Now images of faces will be mapped to points on the half circle manifold corresponding to θ, and non-face images will be mapped to points in the rest of the two dimensional space. Having lots of free space to represent non-face images may be necessary, due to the considerable amount of variability in non-face images. So increasing the dimension of embedding might help us in better separation of face and non-face images (the second criterion). In the case of single pose parameter we suggest 3D embedding.

To make the projection and parameter estimation simple (the third criterion), we embed this half-circle in a three-dimensional space using three equally-spaced shifted cosine functions (Figure 3):

$$F_i(\theta) = \cos(\theta - \alpha_i); \quad i = 1,2,3; \quad \theta = [-\frac{\pi}{2}, \frac{\pi}{2}]; \quad \alpha = \{-\frac{\pi}{3}, 0, \frac{\pi}{3}\}.$$

A point on the face manifold parameterized by the yaw angle θ is $F(\theta) = [F_1(\theta), F_2(\theta), F_3(\theta)]$. When we run the network on an image X, it outputs a vector $G(X)$. The yaw angle $\overline{\theta}$ corresponding to the point on the manifold that is closest to $G(X)$ can be expressed analytically as:

$$\overline{\theta} = \arctan \frac{\sum_{i=1}^{3} G_i(X) \cos(\alpha_i)}{\sum_{i=1}^{3} G_i(X) \sin(\alpha_i)}.$$

The point on the manifold closest to $G(X)$ is just $F(\overline{\theta})$.

The function choice is not limited to cosine. However cosines are preferable since they allow computing the pose analytically from the output of the network. Without this property, finding the pose could be an expensive optimization process, or even require the use of a second learning machine.

The same idea can be generalized to any number of pose parameters. Let us consider the set of all faces with yaw in $[-90, 90]$ and roll in $[-45, 45]$. In an abstract way, this set is isomorphic to a portion of a sphere. Consequently, we can represent a point on the face manifold as a function of the two pose parameters by 9 basis functions that are the cross-products of three shifted cosines for one of the angles, and three shifted cosines for the other angle:

$$F_{ij}(\theta, \phi) = \cos(\theta - \alpha_i) \cos(\phi - \beta_j); \quad i, j = 1,2,3.$$

For convenience, we rescale the roll angles to the range $[-90, 90]$ which allows us to set $\beta_i = \alpha_i$. With this parameterization, the manifold has constant curvature, which ensures that the effect of errors will be the same regardless of pose. Given a 9-dimensional output vector from the convolutional network $G_{ij}(X)$, we compute the corresponding yaw and roll angles $\overline{\theta}, \overline{\phi}$ as follows:

$$\begin{array}{rcl@{\qquad}rcl}
cc & = & \sum_{ij} G_{ij}(X) \cos(\alpha_i) \cos(\beta_j); & cs & = & \sum_{ij} G_{ij}(X) \cos(\alpha_i) \sin(\beta_j); \\
sc & = & \sum_{ij} G_{ij}(X) \sin(\alpha_i) \cos(\beta_j); & ss & = & \sum_{ij} G_{ij}(X) \sin(\alpha_i) \sin(\beta_j); \\
\multicolumn{6}{l}{\overline{\theta} = 0.5(atan2(cs + sc, cc - ss) + atan2(sc - cs, cc + ss)) \quad ;} \\
\multicolumn{6}{l}{\overline{\phi} = 0.5(atan2(cs + sc, cc - ss) - atan2(sc - cs, cc + ss)) \quad .}
\end{array}$$

The process can easily be extended to include pitch in addition to yaw and roll, as well as other parameters if necessary.

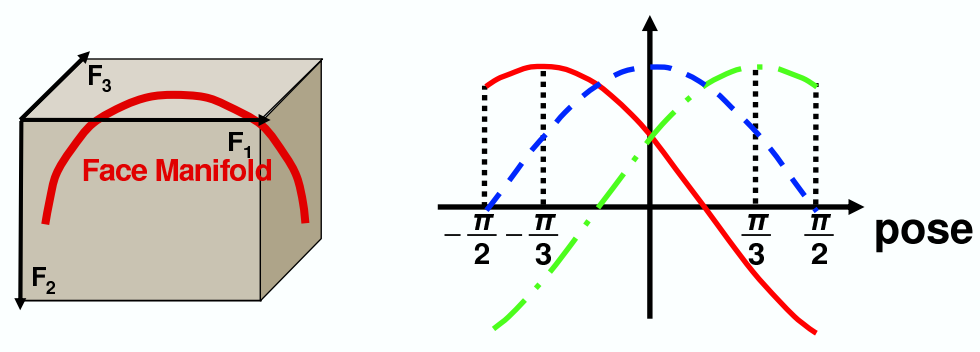

Figure 3: Left: face manifold embedding; right: manifold parametrization by single pose parameter. The value of each cosine function for one pose angle constitute the three components of a point on the face manifold corresponding to that pose.

3. Learning Machine

To map input images to points in the low-dimensional space, we employ a convolutional network architecture trained using Energy Minimization Framework. Next we present the details of the learning machine.

3.1 Energy Minimization Framework

We propose the following configuration of the Energy Based Model (LeCun and Huang, 2005; LeCun et al., 2006). Consider a scalar-valued function $E_W(Y,Z,X)$, where X is a raw image, Z is a facial pose (e.g., yaw and roll as defined above), Y is a binary label: $Y=1$ for face, $Y=0$ for non-face. W is a parameter vector subject to learning. $E_W(Y,Z,X)$ can be interpreted as an *energy function* that measures the degree of compatibility between the values of X,Z,Y. The inference process consists in clamping X to the observed value (the image), and searching for configurations of Z and Y that minimize the energy $E_W(Y,Z,X)$:

$$(\overline{Y},\overline{Z}) = \text{argmin}_{Y\in\{Y\},\, Z\in\{Z\}} E_W(Y,Z,X)$$

where $\{Y\}=\{0,1\}$ and $\{Z\}=[-90,90]\times[-45,45]$ for yaw and roll variables.

Ideally, if the input X is the image of a face with pose Z, then a properly trained system should give a lower energy to the face label $Y=1$ than to the non-face label $Y=0$ for any pose: $E_W(1,Z,X) < E_W(0,Z',X)$, $\forall Z'$. For accurate pose estimation, the system should give a lower energy to the correct pose than to any other pose: $E_W(1,Z',X) > E_W(1,Z,X)$, $\forall Z' \neq Z$. Training a machine to satisfy those two conditions for any image will guarantee that the energy-minimizing inference process will produce the correct answer.

Transforming energies to probabilities can easily be done via Gibbs distribution:

$$P(Y,Z|X) = \exp(-\beta E_W(Y,Z,X)) / \int_{y\in\{Y\},z\in\{Z\}} \exp(-\beta E_W(y,z,X))$$

where β is an arbitrary positive constant, and $\{Y\}$ and $\{Z\}$ are the sets of possible values of y and z. With this formulation, we can easily interpret the energy minimization with respect to Y and Z as

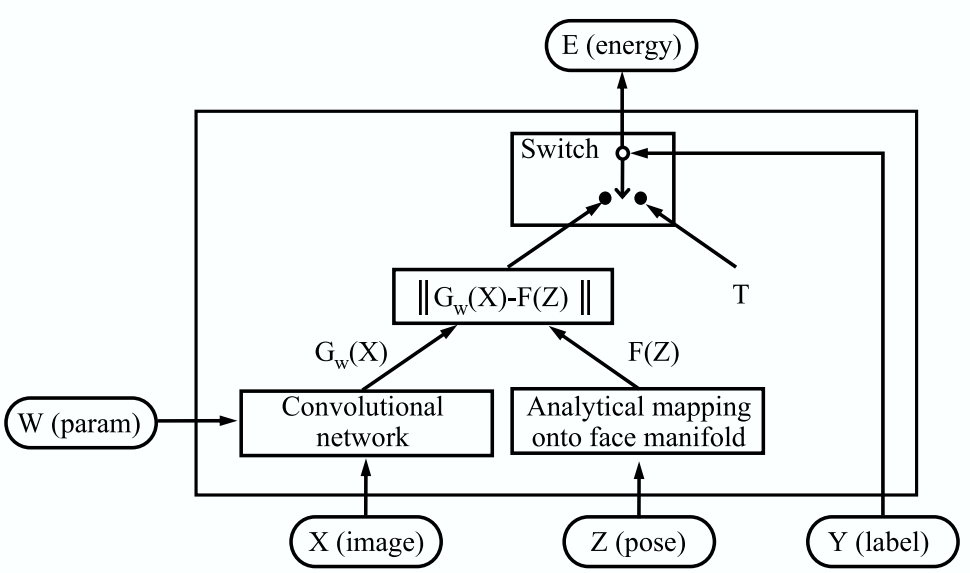

Figure 4: Architecture of the Minimum Energy Machine.

a maximum conditional likelihood estimation of Y and Z. This probabilistic interpretation assumes that the integral in the denominator (the partition function) converges. It is easy to design such an energy function for our case. However, a proper probabilistic formulation would require us to use the negative log-likelihood of the training samples as our loss function for training. This will require us to compute the derivative of the denominator with respect to the trainable parameters W. This is unnecessary complication which can be alleviated by adopting the energy-based formulation. Removing the necessity for normalization gives us complete freedom in the choice of the internal architecture and parameterization of $E_W(Y,Z,X)$, as well as considerable flexibility in the choice of the loss function for training.

Our energy function for a face $E_W(1,Z,X)$ is defined as the distance between the point produced by the network $G_W(X)$ and the point with pose Z on the manifold $F(Z)$:

$$E_W(1,Z,X) = \|G_W(X) - F(Z)\|.$$

The energy function for a non-face $E_W(0,Z,X)$ is equal to a constant T that we can interpret as a threshold (it is independent of Z and X). The complete energy function is:

$$E_W(Y,Z,X) = Y\|G_W(X) - F(Z)\| + (1-Y)T.$$

The architecture of the machine is depicted in Figure 4. Operating this machine (finding the output label and pose with the smallest energy) comes down to first finding: $\overline{Z} = \text{argmin}_{Z\in\{Z\}}\|G_W(X) - F(Z)\|$, and then comparing this minimum distance, $\|G_W(X) - F(\overline{Z})\|$, to the threshold T. If it's smaller than T, then X is classified as a face, otherwise X is classified as a non-face. This decision is implemented in the architecture as a *switch*, that depends upon the binary variable Y.

For simplicity we fix T to be a constant. Although it is also possible to make T a function of pose Z.

3.2 Convolutional Network

We employ a Convolutional Network as the basic architecture for the $G_W(X)$ function that maps image points in the face-space. The architecture of convolutional nets is somewhat inspired by the structure of biological visual systems. Convolutional nets have been used successfully in a number of vision applications such as handwriting recognition (LeCun et al., 1989, 1998), and generic object recognition (LeCun et al., 2004). Several authors have advocated the use of Convolutional Networks for object detection (Vaillant et al., 1994; Nowlan and Platt, 1995; LeCun et al., 1998; Garcia and Delakis, 2002).

Convolutional networks are "end-to-end" trainable system that can operate on raw pixel images and learn low-level features and high-level representation in an integrated fashion. Each layer in a convolutional net is composed units organized in planes called feature maps. Each unit in a feature map takes inputs from a small neighborhood within the feature maps of the previous layer. Neighboring units in a feature map are connected to neighboring (possibly overlapping) windows. Each unit computes a weighted sum of its inputs and passes the result through a sigmoid saturation function. All units within a feature map share the same weights. Therefore, each feature map can be seen as convolving the feature maps of the previous layers with small-size kernels, and passing the sum of those convolutions through sigmoid functions. Units in a feature map detect local features at all locations on the previous layer.

Convolutional nets are advantageous because they can operate on raw images and can easily learn the type of shift-invariant local features that are relevant to image recognition. Furthermore, they are very efficient computationally for detection and recognition tasks involving a sliding window over large images (Vaillant et al., 1994; LeCun et al., 1998).

The network architecture used for training is shown in Figure 5. It is similar to LeNet5 (LeCun et al., 1998), but contains more feature maps. The network input is a 32×32 pixel gray-scale image. The first layer C1 is a convolutional layer with 8 feature maps of size 28×28. Each unit in each feature map is connected to a 5×5 neighborhood in the input. Contiguous units in C1 take input from neighborhood on the input that overlap by 4 pixels. The next layer, S2, is a so-called subsampling layer with 8 feature maps of size 14×14. Each unit in each map is connected to a 2×2 neighborhood in the corresponding feature map in C1. Contiguous units in S2 take input from contiguous, non-overlapping 2x2 neighborhoods in the corresponding map in C1. C3 is convolutional with 20 feature maps of size 10×10. Each unit in each feature map is connected to several 5×5 neighborhoods at identical locations in a subset of S2's feature maps. Different C3 maps take input from different subsets of S2 to break the symmetry and to force the maps to extract different features. S4 is a subsampling layer with 2×2 subsampling ratios containing 20 feature maps of size 5×5. Layer C5 is a convolutional layer with 120 feature maps of size 1×1 with 5×5 kernels. Each C5 map takes input from all 20 of S4's feature maps. The output layer has 9 outputs (since the face manifold is nine-dimensional) and is fully connected to C5 (such a full connection can be seen as a convolution with 1×1 kernels).

3.3 Training with a Contrastive Loss Function

The vector W contains all 63,493 weights and kernel coefficients in the convolutional network. They are all subject to training by minimizing a single loss function. A key element, and a novel contribution, of this paper is the design of the loss function.

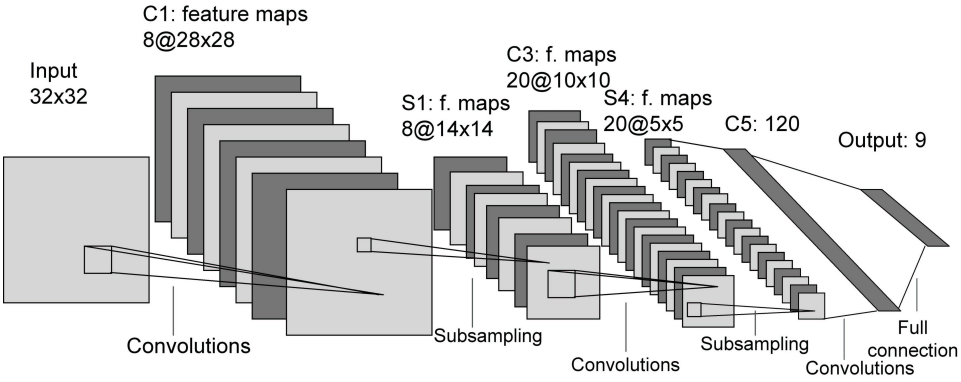

Figure 5: Architecture of convolutional network used for training. This represents one slice of the network with with a 32×32 input window. The slice includes all the elements that are necessary to compute a single output vector. The trained network is replicated over the full input image, producing one output vector for each 32×32 window stepped every 4 pixels horizontally and vertically. The process is repeated at multiple scales.

The training set $\mathcal{S}$ is composed of two subsets: the set $\mathcal{S}_1$ of training samples $(1, X^i, Z^i)$ containing a face annotated with the pose; and the set $\mathcal{S}_0$ of training sample $(0, X^i)$ containing a non-face image (background). The loss function $\mathcal{L}(W, \mathcal{S})$ is defined as the average over $\mathcal{S}_1$ of a per-sample loss function $L_1(W, Z^i, X^i)$, plus the average over $\mathcal{S}_0$ of a per-sample loss function $L_0(W, X^i)$:

$$\mathcal{L}(W, \mathcal{S}) = \frac{1}{|\mathcal{S}_1|} \sum_{i \in \mathcal{S}_1} L_1(W, Z^i, X^i) + \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} L_0(W, X^i). \tag{1}$$

Face samples whose pose is unknown can easily be accommodated by viewing Z as a "deterministic latent variable" over which the energy must be minimized. However, experiments reported in this paper only use training samples manually labeled with the pose.

For a particular positive training sample $(X^i, Z^i, 1)$, the per-sample loss L_1 should be designed in such a way that its minimization with respect to W will make the energy of the correct answer lower than the energies of all possible incorrect answers. Minimizing such a loss function will make the machine produce the right answer when running the energy-minimizing inference procedure. We can write this condition as:

Condition 1

$$E_W(Y^i = 1, Z^i, X^i) < E_W(Y, Z, X^i) \text{ for } Y \neq Y^i \text{ or } Z \neq Z^i.$$

Satisfying this condition can be done by satisfying the two following conditions

Condition 2

$$E_W(1, Z^i, X^i) < T \text{ and } E_W(1, Z^i, X^i) < \min_{Z \neq Z^i} E_W(1, Z, X^i).$$

Following LeCun and Huang (2005), we assume that the loss is a functional that depends on X only through the set of energies associated with X and all the possible values of Z and Y. This assumption allows us to decouple the design of the loss function from the internal structure (architecture) of the energy function. We also assume that there exist a W for which condition 2 is satisfied. This is a reasonable assumption, we merely ensure that the learning machine can produce the correct output for any single sample. We now show that, with our architecture, if we choose L_1 to be a strictly *monotonically increasing* function of $E_W(1,Z^i,X^i)$ (over the domain of E), then minimizing L_1 with respect to W will cause the machine to satisfy condition 2. The first inequality in 2 will obviously be satisfied by minimizing such a loss. The second inequality will be satisfied if $E_W(1,Z,X^i)$ has a single (non-degenerate) global minimum as a function of Z. The minimization of L_1 with respect to W will place this minimum at Z^i, and therefore will ensure that all other values of Z will have higher energy. Our energy function $E_W(1,Z,X) = ||G_W(X) - F(Z)||$ indeed has a single global minimum as a function of Z, because $F(Z)$ is injective and the norm is convex. The single global minimum is attained for $G_W(X) = F(Z)$. For our experiments, we simply chose:

$$L_1(W,1,Z,X) = E_W(1,Z,X)^2.$$

For a particular negative (non-face) training sample $(X^i,0)$, the per-sample loss L_0 should be designed in such a way that its minimization with respect to W will make the energy for $Y=1$ and any value of Z higher than the energy for $Y=0$ (which is equal to T). Minimizing such a loss function will make the machine produce the right answer when running the energy-minimizing inference procedure. We can write the condition for correct output as:

Condition 3

$$E_W(1,Z,X^i) > T \quad \forall Z$$

which can be re-written as:

Condition 4

$$E_W(1,\overline{Z},X^i) > T \quad \overline{Z} = \text{argmin}_z E_W(1,z,X^i).$$

Again, we assume that there exists a W for which condition 4 is satisfied. It is easy to see that, with our architecture, if we choose L_0 to be a strictly *monotonically decreasing* function of $E_W(1,\overline{Z},X^i)$ (over the domain of E), then minimizing L_0 with respect to W will cause the machine to satisfy condition 4. For our experiments, we simply chose:

$$L_0(W,0,X^i) = K \exp[-E(1,\overline{Z},X^i)]$$

where K is a positive constant. A nice property of this function is that it is bounded below by 0, and that its gradient vanishes we approach the minimum.

The entire system was trained by minimizing average value of the loss function in Eq. (1) with respect to the parameter W. We used a stochastic version of the Levenberg-Marquardt algorithm with diagonal approximation of the Hessian (LeCun et al., 1998).

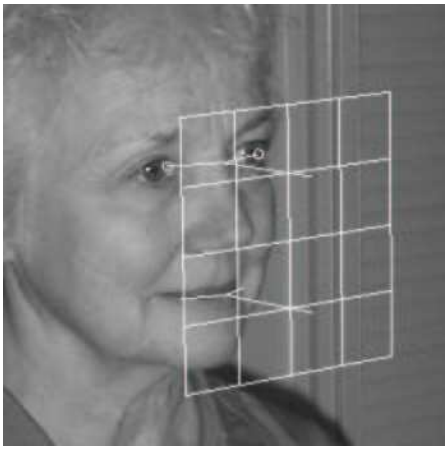

Figure 6: Screenshot from annotation tool.

3.4 Running the Machine

The detection system operates on raw grayscale images. The convolutional network is applied to all 32×32 sub-windows of the image, stepped every 4 pixels horizontally and vertically. Because the layers are convolutional, applying two replicas of the network in Figure 5 to two overlapping input windows leads to a considerable amount of redundant computation. Eliminating the redundant computation yields a dramatic speed up: each layer of the convolutional network is extended so as to cover the entire input image. The output is also replicated the same way. Due to the two 2×2 subsampling layers, we obtain one output vector every 4×4 pixels.

To detect faces in a size-invariant fashion, the network is applied to multiple down-scaled versions of the image over a range of scales stepped by a factor of $\sqrt{2}$. At each scale and location, the network's 9-dimensional output vector is compared to the closest point on the face manifold (whose position indicates the pose of the candidate face). The system collects a list of all locations and scales of output vectors closer to the face manifold than the detection threshold. After examining all scales, the system identifies groups of overlapping detections in the list and discards all but the strongest (closest to the manifold) from each group within an exclusion area of a preset size. No attempt is made to combine detections or apply any voting scheme.

4. Experiments and Results

Using the architecture described in Section 3, we built a detector to locate faces and estimate two pose parameters: yaw from left to right profile, and in-plane rotation from -45 to 45 degrees. The machine was trained to be robust against pitch variation.

In this section, we first describe the training protocol for this network, and then give the results of two sets of experiments. The first set of experiments tests whether training for the two tasks together improves performance on both. The second set allows comparisons between our system and other published multi-view detectors.

4.1 Training

The images we used for training were collected at NEC Labs. All face images were manually annotated with appropriate poses. The annotation process was greatly simplified by using a simple tool for specifying a location and approximate pose of a face. The user interface for this tool is shown in Figure 6. Annotation process is done by first clicking on the midpoint between the eyes and on the center of the mouth. The tool then draws a perspective grid in front of the face and the user adjusts it to be parallel to the face plane. This process yields estimates for all six pose parameters: location (x,y), three angles (yaw, pitch, and roll) and scale. The images were annotated in such a way that the midpoint between the eyes and on the center of the mouth are positioned in the center of the image. This allows these two points to stay fixed when the pose changes from left to right profile. The downside is that profiles occupy only half of the image.

In this manner we annotated about 30,000 faces in images from various sources. Each face was then cropped and scaled so that the eye midpoint and the mouth midpoint appeared in canonical positions, 10 pixels apart, in 32×32-pixel image with some moderate variation of location and scale. The resulting images where mirrored horizontally, to yield roughly 60,000 faces. We removed some portion of faces from this set to yield a roughly uniform distribution of poses from left profile to right profile. Unfortunately, the amount of variation in pitch (up/down) was not sufficient to do the same. This was the reason for training our system to be robust against pitch variation instead of estimating the pitch angle. The roll variation was added by randomly rotating the images in the range of $[-45,45]$ degrees. The resulting training set consisted of $52,850$, 32x32 grey-level images of faces with uniform distribution of poses.

The initial set of negative training samples consisted of $52,850$ image patches chosen randomly from non-face areas in a variety of images. For the second set of tests, half of these images were replaced with image patches obtained by running the initial version of the detector on the training images and collecting false detections.

Each training image was used 5 times during training, with random variations in scale (from $x\sqrt{2}$ to $x(1+\sqrt{2})$), in-plane rotation ($\pm 45^\circ$), brightness (± 20), and contrast (from 0.8 to 1.3).

To train the network, we made 9 passes through this data, though it mostly converged after about the first 6 passes. The training system was implemented in the Lush language (Bottou and LeCun, 2002). The total training time was about 26 hours on a 2Ghz Pentium 4. At the end of training, the network had converged to an equal error rate of 5% on the training data and 6% on a separate test set of 90,000 images.

A standalone version of the detection system was implemented in the C language. It can detect, locate, and estimate the pose of faces that are between 40 and 250 pixels high in a 640×480 image at roughly 5 frames per second on a 2.4GHz Pentium 4.

4.2 Synergy Tests

The goal of the synergy test was to verify that both face detection and pose estimation benefit from learning and running in parallel. To test this claim we built three networks with almost identical architectures, but trained to perform different tasks. The first one was trained for simultaneous face detection and pose estimation (combined), the second was trained for detection only and the third for pose estimation only. The "detection only" network had only one output for indicating whether or not its input was a face. The "pose only" network was identical to the combined network, but trained on faces only (no negative examples). Figure 7 shows the results of running these networks

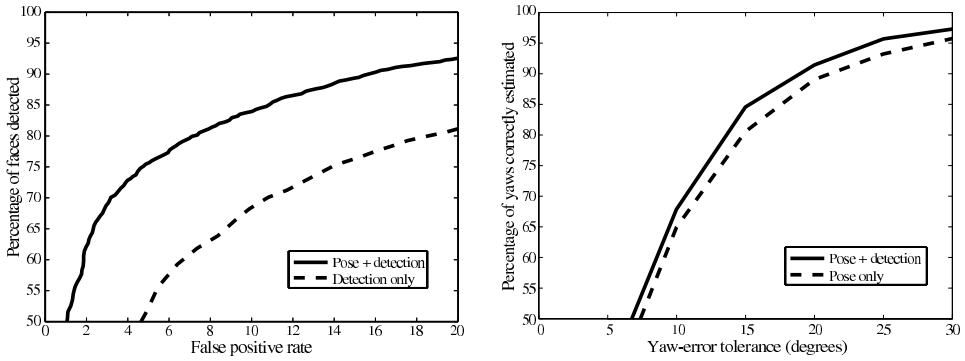

Figure 7: Synergy test. Left: ROC curves for the pose-plus-detection and detection-only networks. (The x axis is the false positive rate per image). Right: frequency with which the pose-plus-detection and pose-only networks correctly estimated the yaws within various error tolerances.

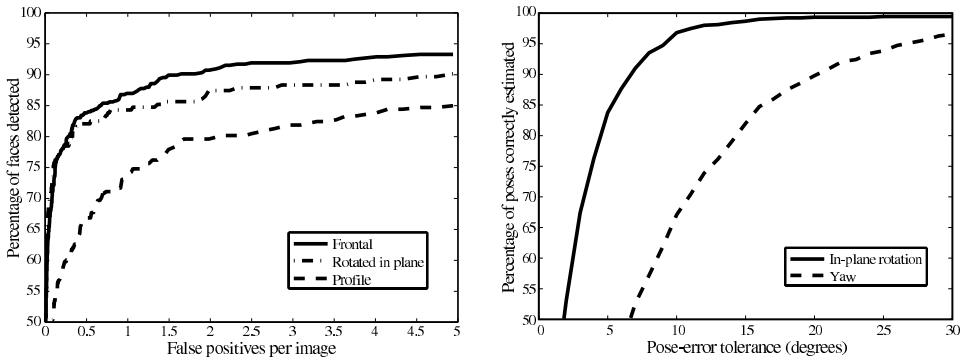

Figure 8: Results on standard data sets. Left: ROC curves for our detector on the three data sets. The x axis is the average number of false positives per image over all three sets, so each point corresponds to a single detection threshold. Right: frequency with which yaw and roll are estimated within various error tolerances.

on our 10,000 test images. In both these graphs, we see that the pose-plus-detection network had better performance, confirming that training for each task benefits the other.

4.3 Standard Data Sets

There is no standard data set that spans the range of poses our system is designed to handle. There are, however, data sets that have been used to test more restricted face detectors, each set focusing on a particular variation in pose. By testing a single detector with all of these sets, we can compare our performance against published systems. As far as we know, we are the first to publish results for a single detector on all these data sets. The details of these sets are described below:

• MIT+CMU (Sung and Poggio, 1998; Rowley et al., 1998a) – 130 images for testing frontal face

detectors. We count 517 faces in this set, but the standard tests only use a subset of 507 faces, because 10 faces are in the wrong pose or otherwise not suitable for the test. (Note: about 2% of the faces in the standard subset are badly-drawn cartoons, which we do not intend our system to detect. Nevertheless, we include them in the results we report.)

• TILTED (Rowley et al., 1998b) – 50 images of frontal faces with in-plane rotations. 223 faces out of 225 are in the standard subset. (Note: about 20% of the faces in the standard subset are outside of the $\pm 45^\circ$ rotation range for which our system is designed. Again, we still include these in our results.)

• PROFILE (Schneiderman and Kanade, 2000) – 208 images of faces in profile. There seems to be some disagreement about the number of faces in the standard set of annotations: Schneiderman and Kanade (2000) reports using 347 faces of the 462 that we found, Jones and Viola (2003) reports using 355, and we found 353 annotations. However, these discrepancies should not significantly effect the reported results.

We counted a face as being detected if 1) at least one detection lay within a circle centered on the midpoint between the eyes, with a radius equal to 1.25 times the distance from that point to the midpoint of the mouth, and 2) that detection came at a scale within a factor of two of the correct scale for the face's size. We counted a detection as a false positive if it did not lie within this range for any of the faces in the image, including those faces not in the standard subset.

The left graph in Figure 8 shows ROC curves for our detector on the three data sets. Figures 9, 10 show detection results on various poses. Table 1 shows our detection rates compared against other multi-view systems for which results were given on these data sets. We want to stress here that all these systems are tested in a pose specific manner: for example, a detector tested on TILTED set is trained only on frontal tilted faces. Such a detector will not be able to detect non frontal tilted faces. Combining all pose variations in one system obviously will increase the number of false positives, since false positives of view-based detectors are not necessarily correlated. Our system is designed to handle all pose variations. This makes the comparison in Table 1 somewhat unfair to our system, but we don't see any other way of comparison against other systems.

The table shows that our results on the TILTED and PROFILE sets are similar to those of the two Jones & Viola detectors, and even approach those of the Rowley *et al* and Schneiderman & Kanade non-real-time detectors. Those detectors, however, are not designed to handle all variations in pose, and do not yield pose estimates. More recent system reported in Huang et al. (2004) is also real-time and can handle all pose variation, but doesn't yield pose estimates. Unfortunately, they also report the results of pose specific detectors. These results are not shown in Table 1, because they report different points on ROC curve in the PROFILE experiment (86.2% for 0.42 f.p per image) and they didn't test on the TILTED set. Even though they trained a combined detector for all pose variations, they did not test it the way we did. Their test consists in running the full detector on the PROFILE set rotated by [-30, 30] degrees in-plane. Unfortunately, they do not provide enough details to recreate their test set.

The right side of Figure 8 shows our performance at pose estimation. To make this graph, we fixed the detection threshold at a value that resulted in about 0.5 false positives per image over all three data sets. We then compared the pose estimates for all detected faces (including those not in the standard subsets) against our manual pose annotations. Note that this test is more difficult than typical tests of pose estimation systems, where faces are first localized by hand. When we hand-localize these faces, 89% of yaws and 100% of in-plane rotations are correctly estimated to within 15°.

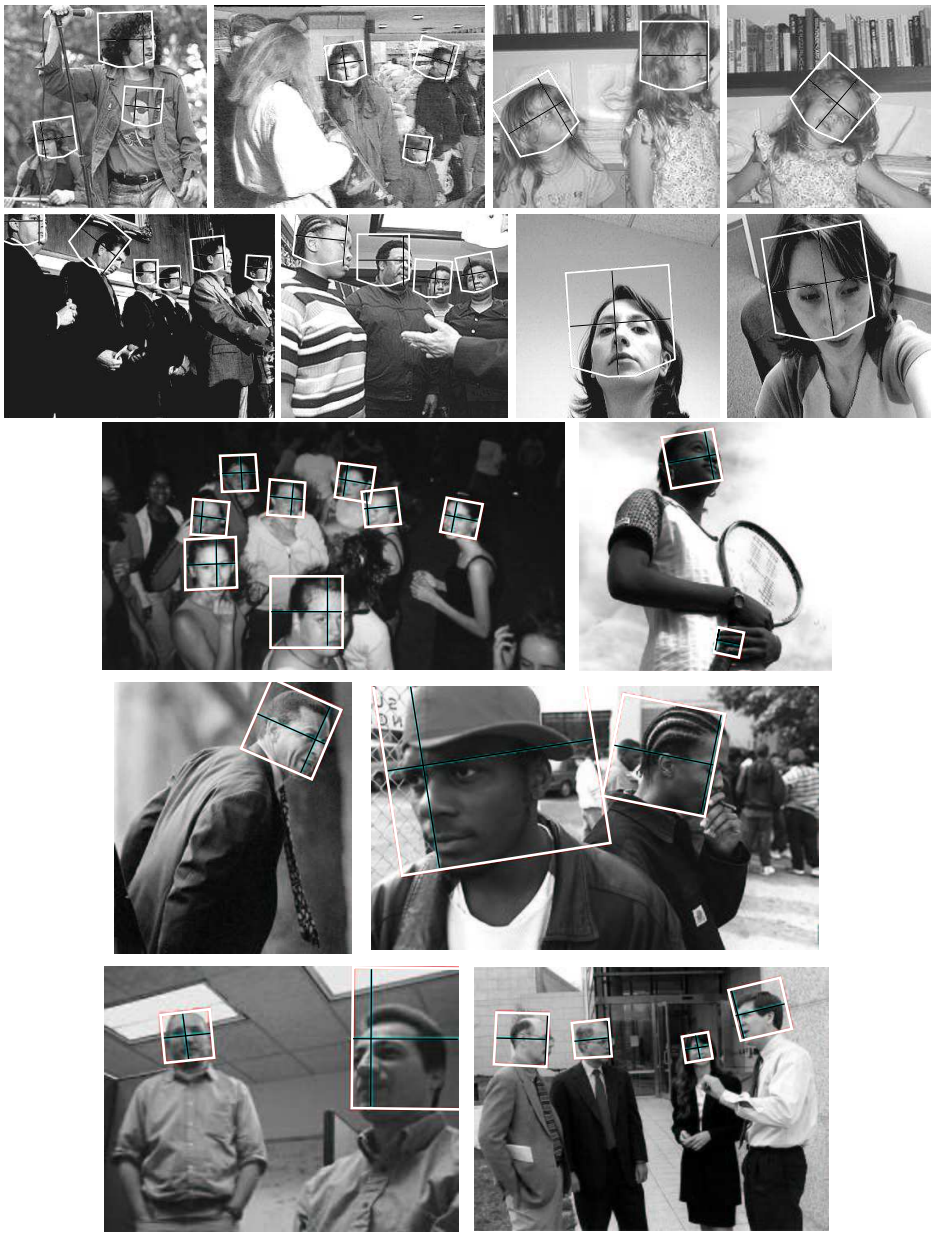

Figure 9: Some example face detections. Each white box shows the location of a detected face. The angle of each box indicates the estimated in-plane rotation. The black crosshairs within each box indicate the estimated yaw.

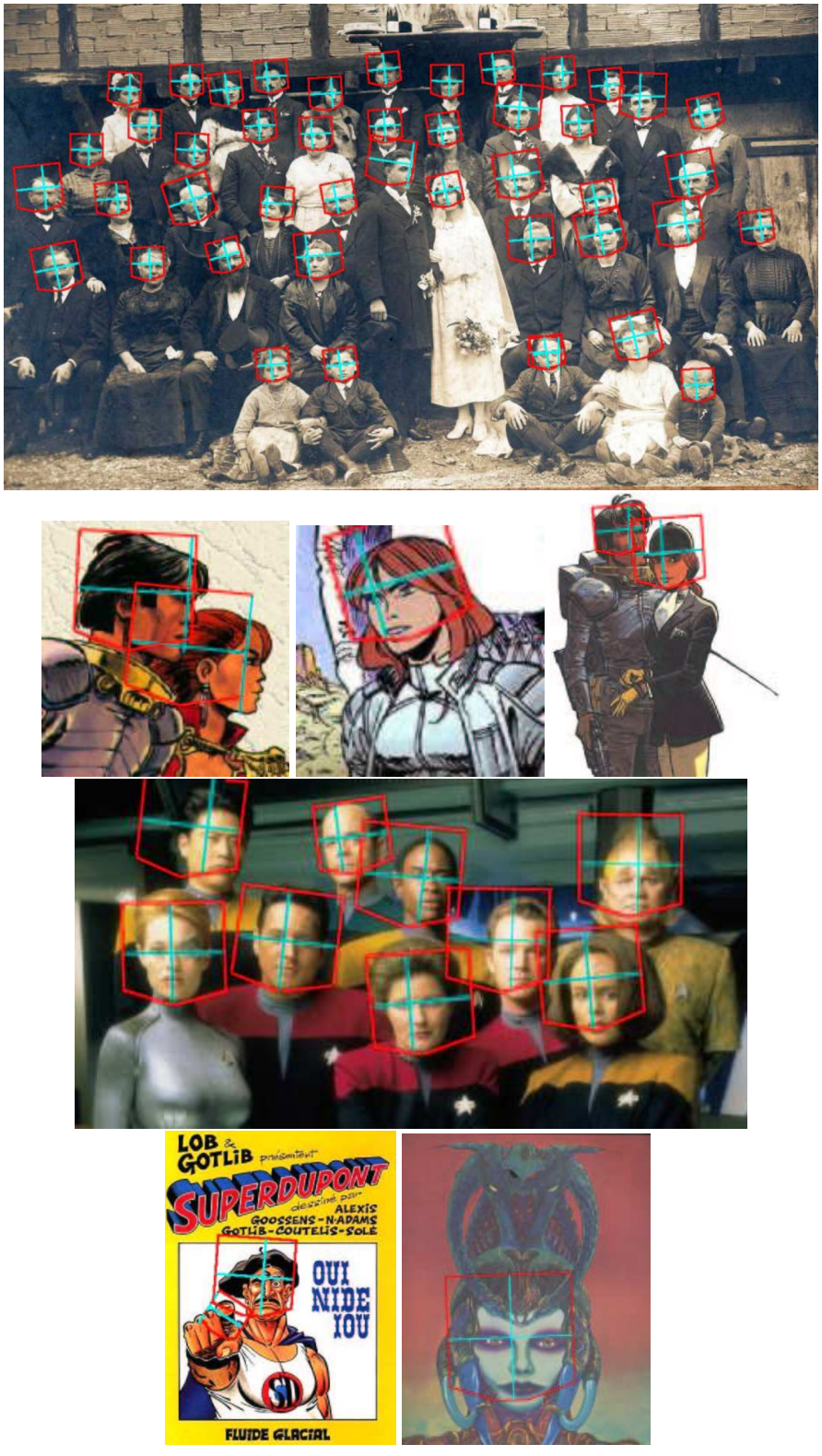

Figure 10: More examples of face detections.

Data set →	TILTED		PROFILE		MIT+CMU	
False positives per image →	4.42	26.90	.44	3.36	.50	1.28
Our detector	**90%**	**97%**	**67%**	**83%**	**83%**	**88%**
Jones and Viola (2003) (tilted)	**90%**	**95%**	x		x	
Jones and Viola (2003) (profile)	x		**70%**	**83%**	x	
Rowley et al. (1998a)	89%	96%	x		x	
Schneiderman and Kanade (2000)	x		86%	93%	x	

Table 1: Comparisons of our results with other multi-view detectors. Each column shows the detection rates for a given average number of false positives per image (these rates correspond to those for which other authors have reported results). Results for real-time detectors are shown in bold. Note that ours is the only single detector that can be tested on all data sets simultaneously.

5. Conclusion

The system we have presented here integrates detection and pose estimation by training a convolutional network to map faces to points on a manifold, parameterized by pose, and non-faces to points far from the manifold. The network is trained by optimizing a loss function of three variables—image, pose, and face/non-face label. When the three variables match, the energy function is trained to have a small value, when they do not match, it is trained to have a large value.

This system has several desirable properties:

• The use of a convolutional network makes it fast. At typical webcam resolutions, it can process 5 frames per second on a 2.4Ghz Pentium 4.

• It is robust to a wide range of poses, including variations in yaw up to $\pm 90^\circ$, in-plane rotation up to $\pm 45^\circ$, and pitch up to $\pm 60^\circ$. This has been verified with tests on three standard data sets, each designed to test robustness against a single dimension of pose variation.

• At the same time that it detects faces, it produces estimates of their pose. On the standard data sets, the estimates of yaw and in-plane rotation are within 15° of manual estimates over 80% and 95% of the time, respectively.

We have shown experimentally that our system's accuracy at *both* pose estimation and face detection is increased by training for the two tasks together.

References

L. Bottou and Y. LeCun. *The Lush Manual*. `http://lush.sf.net`, 2002.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

F. Fleuret and D. Geman. Coarse-to-fine face detection. *IJCV*, pages 85–107, 2001.

C. Garcia and M. Delakis. A neural architecture for fast and robust face detection. *IEEE-IAPR Int. Conference on Pattern Recognition*, pages 40–43, 2002.

C. Huang, B. Wu, H. Ai, and S. Lao. Omni-directional face detection based on real adaboost. In *International Conference on Image Processing*, Singapore, 2004.

M. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-96, Mitsubishi Electric Research Laboratories, 2003.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Winter 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

Y. LeCun, R. Hadsell, S. Chopra, F.-J. Huang, and M.-A. Ranzato. A tutorial on energy-based learning. In *Predicting Structured Outputs*. Bakir et al. (eds),MIT Press, 2006.

Y. LeCun and F. J. Huang. Loss functions for discriminative training of energy-based models. In *Proc. of the 10-th International Workshop on Artificial Intelligence and Statistics (AIStats'05)*, 2005.

Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of CVPR'04*. IEEE Press, 2004.

S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, 2002.

Y. Li, S. Gong, and H. Liddell. Support vector regression and classification based multi-view face detection and recognition. In *Face and Gesture*, 2000.

S. Nowlan and J. Platt. A convolutional neural network hand tracker. In *Advances in Neural Information Processing Systems (NIPS 1995)*, pages 901–908, San Mateo, CA, 1995. Morgan Kaufmann.

M. Osadchy, M. Miller, and Y. LeCun. Synergistic face detection and pose estimation with energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2004)*. MIT Press, 2005.

A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *CVPR*, 1994.

H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *PAMI*, 20:22–38, 1998a.

H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *Computer Vision and Pattern Recognition*, 1998b.

H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Computer Vision and Pattern Recognition*, 2000.

K. Sung and T. Poggio. Example-based learning of view-based human face detection. *PAMI*, 20: 39–51, 1998.

R. Vaillant, C. Monrocq, and Y. LeCun. Original approach for the localisation of objects in images. *IEE Proc on Vision, Image, and Signal Processing*, 141(4):245–250, August 1994.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

M.-H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *PAMI*, 24(1):34–58, 2002.