Face Quality Assessment for Face Recognition in The Surveillance Scenario

A Report

Submitted in partial fulfillment of Summer Internship

For the Degree of

Bachelor of Technology

in the Electronics and Communication Engineering

by

Anil Kumar Vadathya

Roll No: B091877

Rajiv Gandhi University of Knowledge Technologies

Basar – 504107

June - August, 2014

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
आयुध निर्माणि एस्टेट, येद्दुमैलारम ५०२ २०५, आंध्रप्रदेश, भारत.
फोन : +९१-४०-२३०१ ६०३३, फेक्स : +९१-४०-२३०१ ६००३ / ३२
Indian Institute of Technology Hyderabad
Ordnance Factory Estate,
Yeddumailaram 502205, Andhra Pradesh, INDIA
Phone: (040) 2301 6033; Fax: (040) 2301 6003 / 32

CERTIFICATE

My name is Sumohana Channappayya and I am an Assistant Professor of Electrical Engineering at IIT Hyderabad. This is to certify that the summer internship project report on "Face Quality Assessment for Face Recognition in The Surveillance Scenario" is the bonafide work of V. Anil Kumar, Roll No: B091877, 3rd Year B.Tech in Electronics & Communications Engineering (ECE) of IIIT Basar campus of Rajiv Gandhi University of Knowledge Technologies(RGUKT), Telangana carried out under my supervision during 05/06/2014 to 05/08/2014.

Place: IIT Hyderabad
Date : 04/08/2014

Sincerely,

[illegible]

Sumohana Channappayya

INDIAN INSTITUTE OF TECHNOLOGY HYDERABAD
भारतीय प्रौद्योगिकी संस्थान हैदराबाद

Acknowledgements

I thank my Guide Sumohana sir firstly for his selection process; and giving me a chance to fulfill my internship objective to explore the applications of machine learning in signal processing and also the real time applications that can be developed. I also thank K. Sri Rama Murthy sir for his help with the coding of Gaussian Binary RBMs and discussions on results of all the trials.

I thank all the students of *LFoVIA* (Lab For Video and Image Analysis) for giving me a comfortable place and for their help and guidance in all the ways.

I thank Chakravarthi sir for introducing me to this wonderful field of *Machnine Intelligence* and providing me an insight into the research and also Madhu sir for arousing basic questions about convolution which has helped me a lot.

I thank *IIT Hyderabad*, for all the facilities that it has provided especially internet, bus rides and the fun by graduate students. I also thank my intern-mates Shiva, Nitish, Rohit and Naveen for their company and making my stay@iith a great moment.

I would like to thank RGUKT, my university, for giving me this wonderful opportunity as a part of academics and having me equipped with all the necessary knowledge. I thank my dear seniors of Robo*G* group and BMO group for their help in my hunt for internships.

Last but not the least My friends and family for their support all through this way. I would be very happy if someone wants to discuss and give some suggestions regarding this work at anil.rgukt@gmail.com

Abstract

In video based face recognition, face images are typically captured over multiple frames in uncontrolled conditions, where head pose, illumination, shadowing, motion blur and focus change over the sequence. Additionally, inaccuracies in face localisation can also introduce scale and alignment variations. Using all face images, including images of poor quality, can actually degrade face recognition performance. While one solution it to use only the best subset of images, current face selection techniques are incapable of simultaneously handling all of the above mentioned issues. Here in this work, we want to decide whether a detected face is suitable for face recognition or not i.e Face Quality Assessment (FQA), for this purpose we trained a Gaussian Binary RBM with patches (non-overlapping) sampled from good face images. Patches are preprocessed in two different ways; Patch Normalized and ZCA whitened. Machine is doing same reconstructions for faces and non-faces hence we can't use Reconstruction MSE for assessment. Next we used Free-Energy of patches for assessment and observed from free energy images that it is capturing facial features based on the variance. It is assigning free-energy in proportion to the amount of variance in the patch. When this is applied for the trivial case of distinguishing faces and non-faces it is merely checking for the amount of variance as a measure; so faces with high variance (due to beard, glasses and expressions) overlapped with non-faces and similarly non-faces with smooth regions overlapped with the faces. In all this training we have been training a single rbm to learn all the local features at so small scale that is able to reconstruct natural images in general. So, with this experience we want to train individual RBMs for face features and combine them to form a big RBM and use it for FQA.

Contents

List of Tables

List of Figures

Notation and Abbreviations

E - Energy Function

V - Number of visible units

H - Number of hidden units

$\mathbf{v}$ - state vector of visible units

$\mathbf{h}$ - state vector of hidden units

v_i - i^{th} visible unit

h_j - j^{th} hidden unit

σ_i - variance of i^{th} visible unit

b_i^v - bias of i^{th} visible unit

b_j^h - bias of j^{th} hidden unit

ϵ_w - learning rate of parameter w

F - Free Energy function

RBM - Restricted Boltzmann Machine

GRBM - Gaussian Restricted Boltzmann Machine

AIS - Annealed Importance Sampling

CD - Contrastive Divergence

PCD - Persistent Contrastive Divergence

rbm50 - rbm with parameters at 50^{th} iteration

RMSE - Reconstruction Mean Square Error

Chapter 1

Introduction

Video-based identity inference in surveillance conditions is challenging due to a variety of factors, including the subjects motion, the uncontrolled nature of the subjects, variable lighting, and poor quality CCTV video recordings. This results in issues for face recognition such as low resolution, blurry images (due to motion or loss of focus), large pose variations, and low contrast [8, 9, 10, 11]. While recent face recognition algorithms can handle faces with moderately challenging illumination conditions [12], strong illumination variations (causing cast shadows [13] and self-shadowing) remain problematic [14].

One approach to overcome the impact of poor quality images is to assume that such images are outliers in a sequence. This includes approaches like exemplar extraction using clustering techniques (eg. k-means clustering [15]) and statistical model approaches for outlier removal [16]. However, these approaches are not likely to work when most of the images in the sequence have poor quality the good quality images would actually be classified as outliers.

Another approach is explicit subset selection, where a *face quality assessment* is automatically made on each image, either to remove poor quality face images, or to select a subset comprised of high quality images [17, 18, 19]. This improves recognition performance, with the additional benefit of reducing the overall computation load during feature extraction and matching. The challenge in this approach is finding a good definition for face quality.

Here in this work we want to do the Face Quality Assessment (FQA) by training a Restricted Boltzmann Machine (RBM) with normal faces and then use it for selecting good quality faces from the set of detected faces.

1.1 Energy based learning models

Boltzmann Machine is a parallel computational organization that is well suited to constraint satisfaction tasks involving large numbers of "weak constraints" [5]. Constraint-satisfaction searches normally use "strong" constraints that must be satisfied by any solutions[7]. In some problem domains, such as finding the most plausible interpretation of an image, many of the criteria are not all-or-none, and frequently even the best possible solution violates some constraints [4]. A variation that is more appropriate for such domains uses weak constraints that incur a cost when violated. The quality of a solution is then determined by the total cost of all the constraints that it violates. In a perceptual interpretation task, for example, this total cost should reflect the implausibility of the interpretation.

The machine is composed of primitive computing elements called *units* that are connected to each other by bidirectional *links*. A unit is always in one of two states, *on* or *off*, and it adopts these states as a probabilistic function of the states of its neighbouring units and the *weights* on its links to them. The weights can take on real values of either sign. A unit being on or off is taken to mean that the system currently accepts or rejects some elemental hypothesis about the domain. The weight on a link represents a weak pairwise constraint between two hypothesis. A positive weight indicates that the two hypotheses tend to support one another; if one is currently accepted, accepting the other should be more likely. Conversely, a negative weights suggests, other things being equal, that the two hypothesis should not both be accepted. Link weights are symmetric, having the same strength in both directions.

The cost for each global state of the machine is assigned as the *energy*(E) of that particular state[6], with rights assumptions, the individual units can be made to act as to *minimize the global energy.* If some of the units are externally forced or "clamped" into particular sates to represent a particular input, the system will then find the minimum energy configuration that is compatible with that input. The energy of a configuration can be interpreted as the extent to which that combination of hypotheses violates the constraints implicit in the problem domain, so in minimizing energy the machine evolves towards "interpretations" of that input that increasingly satisfy the problem domain.
The energy of a global state is defined as

$$E(\mathbf{x}) = -\sum_{i<j} w_{ij} x_i x_j + \sum_i b_i x_i \tag{1.1}$$

where w_{ij} is the strength of connection between units i and j, x_i is 1 if unit i is on and 0 otherwise, and b_i is the bias.
The probability to the global state is defined by the Boltzmann distribution

$$p(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{X})} \tag{1.2}$$

where $\mathbf{x}$ is the state vector of units and Z is called the partition function which a normalization constant and is given by

$$Z = \sum_{\mathbf{X}} e^{-E(\mathbf{X})} \tag{1.3}$$

If the Boltzmann machine is trained with specific patterns then it evolves by assigning low energies and thereby higher probabilities to the trained pattern whereas to others it assigns higher energies and hence low probabilities.

Chapter 2

Restricted Boltzmann Machines

Restricted Boltzmann Machines(RBMs) are computationally tractable versions of the Boltzmann machines which are energy based stochastic learning models where the state of each unit turning *on* or *off* is probabilistically determined by the current state of all other neurons in the network. They can be trained in both *supervised* and *unsupervised* manner[3].

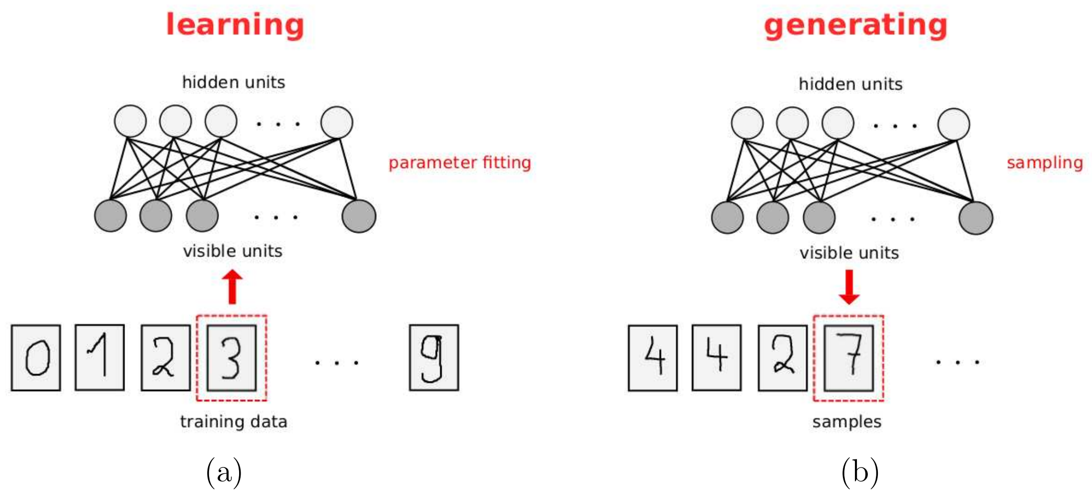


Figure 2.1: (a) Learning an RBM corresponds to fitting its parameters such that the distribution represented by the RBM models the distribution underlying the training data, here handwritten digits. (b) After learning, the trained RBM can be used to generate samples from the learned distribution.

In general, learning a Boltzmann machine is computationally demanding. However, the learning problem can be simplified by imposing restrictions on the network topology. In Boltzmann machines two types of units can be distinguished. They have visible neurons and potentially hidden neurons. Restricted Boltzmann machines always have both types of units, and these can be thought of as being arranged in two layers, see figure 2.1 for an illustration. The visible units constitute the first layer and correspond to the components of an observation (e.g., one visible unit for each pixel of a digital input image). The hidden units model dependencies between the components of observations (e.g., dependencies between the pixels in the images) and can be viewed as non-linear feature detectors [2]. In the RBMs network graph, each neuron is connected to all the neurons in the other layer. However, there are no connections between neurons in the same layer, and this restriction gives the RBM its name.

A Binary-Binary RBM with V visible units and H hidden units is governed by the following energy function:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i w_{ij} h_j - \sum_{i=1}^{V} v_i b_i^v - \sum_{j=1}^{H} h_j b_j^h \tag{2.1}$$

The associated probability with the configuration $(\mathbf{v},\mathbf{h})$ is given by

$$p(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}} \tag{2.2}$$

The sum in the denominator is over all possible visible and hidden configurations, and is thus extremely hard to compute when the number of units is large.
The probability of a particular visible state configuration $\mathbf{v}$ is derived as follows:

$$\begin{aligned} p(\mathbf{v}) &= \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}) \\ &= \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}} \end{aligned} \tag{2.3}$$

At a very high-level, the RBM training procedure consists of fixing the states of the visible units $\mathbf{v}$ at some desired configuration and then finding settings of the parameters (the weights and biases) such that $p(\mathbf{v})$ is large. The hope is that the model will use the hidden units to generalize and to extract meaningful features from the data, and hence $p(\mathbf{u})$ will also be large for some $\mathbf{u}$ drawn from the same distribution as $\mathbf{v}$.

The conditional probabilities are given by

$$\begin{aligned} p(\mathbf{v}|\mathbf{h}) &= \frac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{h})} \\ &= \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u}} e^{-E(\mathbf{u},\mathbf{g})}} \end{aligned} \tag{2.4}$$

$$\begin{aligned} p(\mathbf{h}|\mathbf{v}) &= \frac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{v})} \\ &= \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}} \end{aligned} \tag{2.5}$$

We can also derive a closed-form expression for $p(v_k = 1|\mathbf{h})$, the probability of turning a visible unit on given the hidden units configuration[1]

$$\begin{aligned} p(v_k = 1|\mathbf{h}) &= \frac{p(v_k = 1, \mathbf{v}_{i\neq k}, \mathbf{h})}{p(\mathbf{h})} \\ &= \frac{1}{1 + e^{-\left(\sum_{j=1}^{H} h_j w_{kj} + b_k^v\right)}} \end{aligned} \tag{2.6}$$

In the similar way,

$$p(h_k = 1|\mathbf{v}) = \frac{1}{1 + e^{-\left(\sum_{i=1}^{V} v_i w_{ki} + b_k^h\right)}} \tag{2.7}$$

So as can be expected from Figure 2.1, we find that the probability of a visible unit turning on is independent of the states of the other visible units, given the states of the hidden units. Likewise, the hidden states are independent of each other given the visible states. This property of RBMs makes sampling extremely efficient, as one can sample all the hidden units simultaneously and then all the visible units simultaneously.

[1]see Appendix A for complete derivation

2.1 Gaussian binary RBMs

A GRBM with V visible units and H hidden units; to model real-valued data, we use model's energy function as

$$E(\mathbf{v},\mathbf{h}) = \sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} - \sum_{j=1}^{H} b_j^h h_j - \sum_{i=1}^{V}\sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij} \tag{2.8}$$

This type of model is explored in [20, 21]. Here, v_i denotes the now *real-valued* activity of visible unit v_i. Notice that here each visible unit adds a parabolic (quadratic) offset to the energy function, where σ_i controls the width of parabola. Given the energy function, we can derive the distribution $p(\mathbf{v}|\mathbf{h})$ as follows[2]:

$$\begin{aligned} p(\mathbf{v}|\mathbf{h}) &= \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\int_{\mathbf{u}} e^{-E(\mathbf{u},\mathbf{h})}\, d\mathbf{u}} \\ &= \prod_{i=1}^{V} \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2\sigma_i^2}(v_i - b_i^v - \sigma_i \sum_{j=1}^{H} h_j w_{ij})^2} \end{aligned} \tag{2.9}$$

Which we recognize as the V-dimensional Gaussian distribution with diagonal covariance given by

$$\begin{pmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_V^2 \end{pmatrix}$$

and mean in i^{th} dimension is given by

$$b_i^v + \sigma_i \sum_{j=1}^{H} h_j w_{ij}$$

As before, we can compute $p(h_k|\mathbf{v})$ as follows:

$$p(h_k = 1|\mathbf{v}) = \frac{\sum_{h_{j\neq k}} p(\mathbf{v}, h_k = 1, h_{j\neq k})}{p(\mathbf{v})}$$

[2]see Appendix A for complete derivation

$$= \frac{1}{1+e^{-\left(\sum_{i=1}^{V}\frac{v_i}{\sigma_i}w_{ik}+b_k^h\right)}} \tag{2.10}$$

2.1.1 Training of GRBM

Given the set of C training cases $\{\mathbf{v}^c | c \in \{1, 2, \ldots, C\}\}$, the goal is to maximize the average log probability of the set under the model's distribution:

$$\sum_C \log p(\mathbf{v}^c) = \sum_C \log \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}^c,\mathbf{h})}}{\sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}} \tag{2.11}$$

We attempt to do this with gradient descent. Differentiating with respect to a weight w_{ij}, we have

$$\frac{\partial}{\partial w_{ij}}\sum_C \log p(\mathbf{v}^c) = \frac{\partial}{\partial w_{ij}}\left(\sum_C \log \sum_{\mathbf{h}} e^{-E(\mathbf{v}^c,\mathbf{h})} - \log \sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}\right). \tag{2.12}$$

Now the first term in the equation

$$\begin{aligned}\frac{\partial}{\partial w_{ij}}\sum_C \log \sum_{\mathbf{h}} e^{-E(\mathbf{v}^c,\mathbf{h})} &= -\sum_C \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}^c,\mathbf{h})}\frac{\partial E(\mathbf{v}^c,\mathbf{h})}{\partial w_{ij}}}{\sum_{\mathbf{g}} e^{-E(\mathbf{v}^c,\mathbf{g})}} \\ &= -\frac{1}{\sigma_i}\sum_C \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}^c,\mathbf{h})} v_i^c h_j^c}{\sum_{\mathbf{g}} e^{-E(\mathbf{v}^c,\mathbf{g})}} \\ &= -\frac{1}{\sigma_i}\sum_C \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}^c) v_i^c h_j \end{aligned} \tag{2.13}$$

we can see that last expression is just the expectation of $v_i^c h_j^c$ given that $\mathbf{v}$ is clamped to the data vector $\mathbf{v}^c$. Since we know $\mathbf{v}^c$ we can calculate the expected value[3] of h_j using 2.10 easily.

Coming to the second term,

$$\frac{\partial}{\partial w_{ij}} \log \sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} = -\frac{1}{\sigma_i}\frac{\sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} u_i g_j}{\sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}}$$

[3]Estimation of expectation using samples from the actual distribution
See http://ib.berkeley.edu/labs/slatkin/eriq/classes/guest_lect/mc_lecture_notes.pdf

$$= -\frac{1}{\sigma_i}\sum_{\mathbf{u}}\sum_{\mathbf{g}} p(\mathbf{u},\mathbf{g})u_i g_j \tag{2.14}$$

Here, the expression is expected value of $u_i g_j$ under the model's distribution which is mathematically intractable. We can compute $u_i g_j$ by clamping the visible units to the data vector $\mathbf{v}^c$, then sampling the hidden units, then sampling the visible units using 2.9, and repeating this procedure for infinitely many times. After infinitely many iterations, the model will have forgotten its starting point and we will be sampling from its equilibrium distribution. However, it has been shown in [24] that this expectation can be approximated well in finite time by a procedure known as Contrastive Divergence (CD). The CD learning procedure approximates (1.6) by running the sampling chain for only a few steps. We name CD-N the algorithm that samples the hidden units $N + 1$ times.

In practice, we use CD-1 almost exclusively because it produces adequate results. CD-1 learning amounts to lowering the energy that the model assigns to training vectors and raising the energy of the model's "reconstructions" of those training vectors. CD-1 is fast, has low variance, and is a reasonable approximation to the likelihood gradient, but it is still significantly different from the likelihood gradient when the mixing rate is low, so we are using a variant of it known as Persistent Contrastive Divergence Learning (PCD)[23]. To approximate the expectation we need a sample from the model distribution and for this we use Markov Chain, but running a chain for many steps is too time-consuming. However, between parameter updates, the model changes only slightly. We can take advantage of that by initializing a Markov Chain at the state in which it ended for the previous model. This initialization is often fairly close to the model distribution, even though the model has changed a bit in the parameter update. Of course this still is an approximation, because the model does change slightly with each parameter update. With infinitesimally small learning rate it becomes exact, and in general it seems to work best with small learning rates.

The update rule of weight w_{ij} is

$$\Delta w_{ij} = \epsilon_w \left(E_{data}[v_i h_j] - E_{model}[v_i h_j] \right) \tag{2.15}$$

for biases is

$$\Delta b_i^v = \epsilon_{b^v} \left(E_{data}[v_i] - E_{model}[v_i] \right) \tag{2.16}$$

$$\Delta b_j^h = \epsilon_{b^h} \left(E_{data}[h_j] - E_{model}[h_j] \right) \tag{2.17}$$

where ϵ_w is the weight learning rate hyperparameter, E_{data} is the expectation under the model's distribution when the the visible units are clamped to the data, and E_{model} is the expectation under the model's distribution when the visible units are unclamped. As discussed above, E_{model} is approximated using PCD.

2.1.2 Learning visible variances

We attempt to maximize the log probability of the data vectors

$$\begin{aligned}\log p(\mathbf{v}) &= \log \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\int_{\mathbf{u}} \sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} d\mathbf{u}} \\ &= \log \sum_{h} e^{-E(\mathbf{v},\mathbf{h})} - \log \int_{\mathbf{u}} \sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} d\mathbf{u}\end{aligned} \tag{2.18}$$

The first term is the negative of the free energy that the model assigns to vector $\mathbf{v}$, and it can be expanded as follows:

$$\begin{aligned}-F(\mathbf{v}) &= \log \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \\ &= -\sum_{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} + \sum_{H} \log \left(1 + e^{b_j^h + \sum_V \frac{v_i}{\sigma_i} w_{ij}} \right)\end{aligned} \tag{2.19}$$

The derivative of $-F(\mathbf{v})$ with respect to σ_i is

$$\frac{\partial(-F(\mathbf{v}))}{\partial \sigma_i} = \frac{(v_i - b_i^v)^2}{\sigma_i^3} - \sum_{H} \left(\frac{1}{1 + e^{-b_j^h - \sum_V \frac{v_i}{\sigma_i} w_{ij}}} \right) \cdot \frac{w_{ij} v_i}{\sigma_i^2}$$

$$= \frac{(v_i - b_i^v)^2}{\sigma_i^3} - \sum_H a_j \cdot \frac{w_{ij} v_i}{\sigma_i^2} \tag{2.20}$$

Likewise, the derivative of second term of equation 2.18 with σ_i gives

$$\frac{\partial}{\partial \sigma_i} \log \int_{\mathbf{u}} \sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} d\mathbf{u} = \frac{\int_{\mathbf{u}} \sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} \cdot \left(\frac{(u_i - b_i^v)^2}{\sigma_i^3} - \sum_H g_j \cdot \frac{w_{ij} u_i}{\sigma_i^2} \right) d\mathbf{u}}{\int_{\mathbf{u}} \sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})} d\mathbf{u}} \tag{2.21}$$

which is just the expected value of

$$\frac{(v_i - b_i^v)^2}{\sigma_i^3} - \sum_H h_j \cdot \frac{w_{ij} v_i}{\sigma_i^2} \tag{2.22}$$

under model's distribution. Therefore, the updated rule for σ_i is

$$\Delta \sigma_i = \epsilon_\sigma \left(E_{data} \left[\frac{(v_i - b_i^v)^2}{\sigma_i^3} - \sum_H h_j \cdot \frac{w_{ij} v_i}{\sigma_i^2} \right] - E_{model} \left[\frac{(v_i - b_i^v)^2}{\sigma_i^3} - \sum_H h_j \cdot \frac{w_{ij} v_i}{\sigma_i^2} \right] \right) \tag{2.23}$$

where ϵ_σ is the learning rate hyperparameter. As for weights, we use PCD to approximate the expectation under the model.

2.1.3 Visualizing filters

There is an intuitive way to interpret the weights of a Gaussian-Bernoulli RBM trained on images, and that is to look at the filters that the hidden units apply to the image. Each hidden unit is connected with some weight to each pixel. If we arrange these weights on a 14×14 grid, we obtain a visualization of the filter applied by the hidden unit. Figure 3.4 shows this type of visualization of 300 hidden units. In creating these visualizations, we use intensity to indicate the strength of the weight, where white color indicates the strong positive strengths whereas black indicates the strong negative connections and the values in between are assigned according to the gray-scale.

2.2 Measuring performance

Ideally, we would like to evaluate $p(\mathbf{v})$ for data vector $\mathbf{v}$. But this is intractable for models with large numbers of hidden units due to the second term in eqn. 2.18. Instead we use the "reconstruction error". The reconstruction error is the squared difference $(\mathbf{v} - \mathbf{v}')$ between the data vector $\mathbf{v}$, and the vector produced by sampling the hidden units given $\mathbf{v}$ to obtain a hidden configuration $\mathbf{h}$, and then sampling the visible units given $\mathbf{h}$ to obtain a visible configuration $\mathbf{v}'$. This is not a very good measure of how well the model's probability distribution matches that of the data, because the reconstruction error depends heavily on how fast the Markov chain mixes.

2.3 Preprocessing of Face Images

For Training GRBMs, initially we considered frontal gray face images from the FERET database[1] from which non-overlapping patches of size 14×14 are sampled. The sampled patches are now random vectors of dimension 196 which are preprocessed in two different ways[4]:

1. Patches are normalized for zero mean and unit variance
2. Patches are ZCA whitened

Results of the individual cases are discussed below.

Figure 2.2: Sample resized frontal faces from the Feret database

[4]Pixel intensities are scaled down by 100 for computational ease

2.3.1 Patch Normalization

In this each patch vector $X = \{x_1, x_2, ..., x_N\}$ is normalized to X_{norm} for zero mean and unit variance using eqn 2.1. This normalized patch is used as input to GRBM composed of visible units(n_v) and hidden units(n_h)

$$X_{norm} = \frac{X - \mu_X}{var(X) + 1}; \; where \; \mu_X = \frac{1}{N}\sum_{i=1}^{N} x_i \; and \; var(X) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_X)^2 \quad (2.24)$$

2.3.2 ZCA whitening

In this the N-dimensional patches are whitened i.e transformed such that they are decorrelated, this removes the lower order distributions and enables the machine to learn higher order statistics of data(structure). The ZCA whitening transform is done as follows:

The n N-dimensional patch vectors are arranged into $N \times n$ matrix A, the patch vectors are assumed to have zero mean and their covariance is given by $\frac{1}{n-1}AA^T$, now to decorrelate data dimensions we use a linear transform W as

$$Y = WA$$

In order for W to decorrelate data matrix A, YY^T must be diagonal. However, we can restrict our search only to Ws that satisfy

$$YY^T = (n-1)I$$

There are multiple Ws that satisfy this, so we can restrict further by requiring $W = W^T$. Given these, we can find W:

$$\begin{aligned}
YY^T &= (n-1)I \\
WAA^TW^T &= (n-1)I \\
W^TWAA^TW^T &= (n-1)W^T \\
W^2AA^TW^T &= (n-1)W^T \\
W^2AA^T &= (n-1)I \\
W^2 &= (n-1)(AA^T)^{-1} \\
W &= \sqrt{n-1}(AA^T)^{-1/2} \qquad (2.25)
\end{aligned}$$

$(AA^T)^{-1/2}$ can be found out since AA^T is symmetric and hence orthogonally diagonalizable. That is,

$$AA^T = PDP^T$$

for some P orthogonal matrix and D diagonal matrix. So

$$\begin{aligned} (AA^T)^{-1/2} &= ((AA^T)^{-1})^{1/2} \\ &= ((PDP^T)^{-1})^{1/2} \\ &= (PD^{-1}P^T)^{1/2} \\ &= PD^{-1/2}P^T \qquad (2.26) \end{aligned}$$

So, $W = (XX^T)^{1/2}$ transforms X in such a way that the resultant data dimensions are uncorrelated with one another and the variance in each dimension is exactly 1. W may also be thought of as rotating to the space of its principal components, dividing each principal component by the square root of the variance in that direction, and then rotating back to pixel space. W is called *a whitening matrix*, and is referred to as the *Zero Components Analysis* (ZCA) solution to the equation

$$YY^T = (n-1)I$$

The *dewhitening matrix*, W^{-1}, is given by

$$W^{-1} = PD^{1/2}P^T.$$

Chapter 3

Results and Discussions

3.1 Patch Normalized Images training

As discussed in Preprocessing section normalized 41,688 patches(14×14) are sampled from 193 faces and 11,000 patches from other 50 faces are used as evaluation data for assessing the machine while training. We used Persistent Contrastive Divergence(PCD) for generating negative samples for evaluating the gradient for parameter updates. Algorithmic parameters[1] used are mentioned in the below table:

Table 3.1: GRBM Paramters for Training normalized patches

Parameter	value
nv	196
nh	1200
ϵ	0.01(Decreased Iterwise)
ϵ_σ	0.0001
weightCost	0.001
sparseCost	0.001
sparseTarget	0.1(Decreased to 0.05)
Gibbs cycles	1
batchSize	200
maxIter	1000

While tuning the parameters for training we found that ϵ_σ should be very low when compared to ϵ otherwise the machine diverges suddenly after few iterations and the

[1]Unless mentioned parameters are kept constant for all iterations

variances(σ_i) turnout to be hugely negative. Maximum number of iterations is set to 1000, since the batch size is 200 for 41,688 patches in each iteration parameters are updated for 20,000 times. Although the maximum iterations was set to 1000 we stopped the learning when reconstruction error stopped improving further.

3.1.1 Training Phase

Hidden Units Activation The hidden units activation probabilities at some iterations for a batch are shown below from those we can observe the sparsity effect as they become much darker i.e only hidden units capturing features getting fired

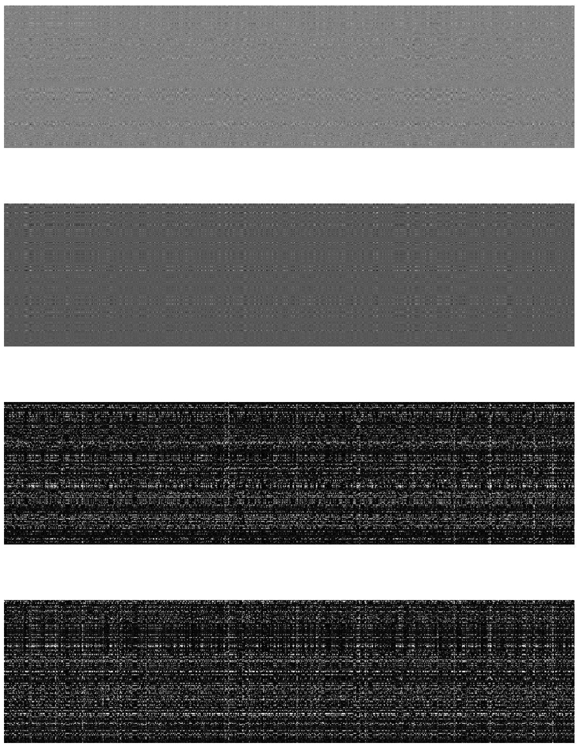


Figure 3.1: Hidden unit activation probabilities for a batch at iter: 1,17,126,486

Error Convergence and Free Energy behavior Initially the error reduced fast (see figure 3.2), as iterations progressed it has settled and slowly it started diverging at this point(490^{th} iteration) we stopped learning. Free energy for the training data has increased rapidly till 100 iterations (see figure 3.3) later it started decreasing very slowly, as per the literature free energy should decrease but it never happened during my trials. The gap between free energies of Training and Evaluating data represents the amount of Overfit, which is very less till 100 iterations later it has increased a bit.

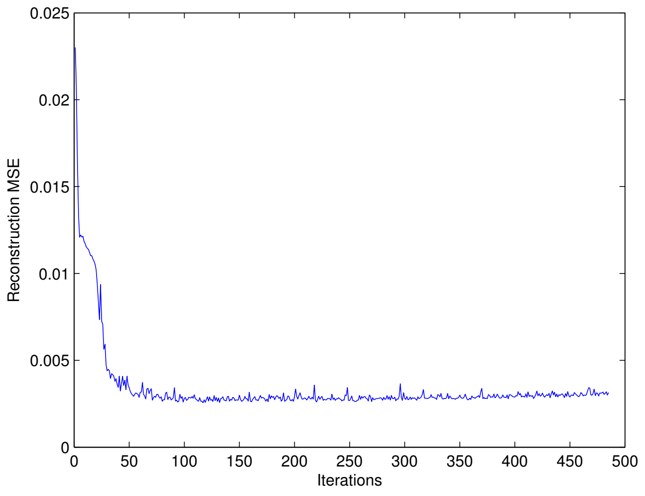


Figure 3.2: Error Vs Iterations

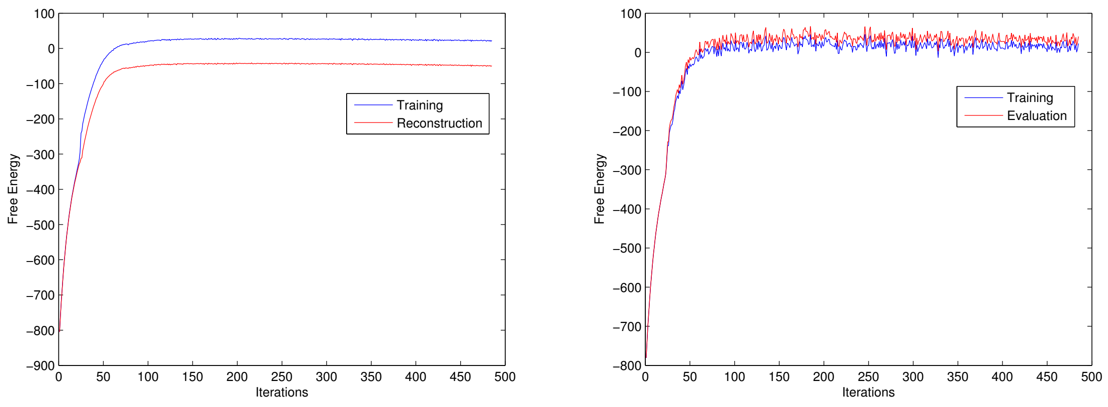


Figure 3.3: Free Energy Vs Iterations

The reconstructions are not much different for 1 Gibbs cycle sampling with rbm50 and rbm480 parameters however there was significant difference for 10 Gibbs cycle samplings with rbm50 and rbm480, this we will discuss in detail in the Reconstructions section below.

3.1.2 Testing and Analysis

Evolution of Weights and Reconstructions As the iterations progress GRBM weights are improving from the smooth (figure 3.4) to edgy filters (figure 3.5) and their modeling capacity has also improved as can be seen from the reconstructions, slowly these filters are turning into —. But with those weights it is even capable of reconstructing

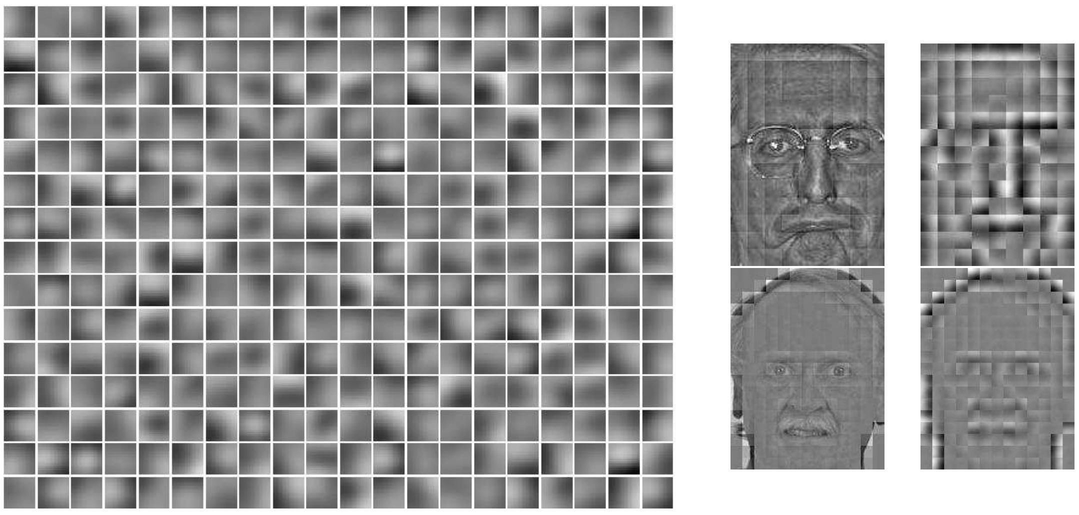


Figure 3.4: Weights of 300 hidden units at 20th iteration and reconstructions

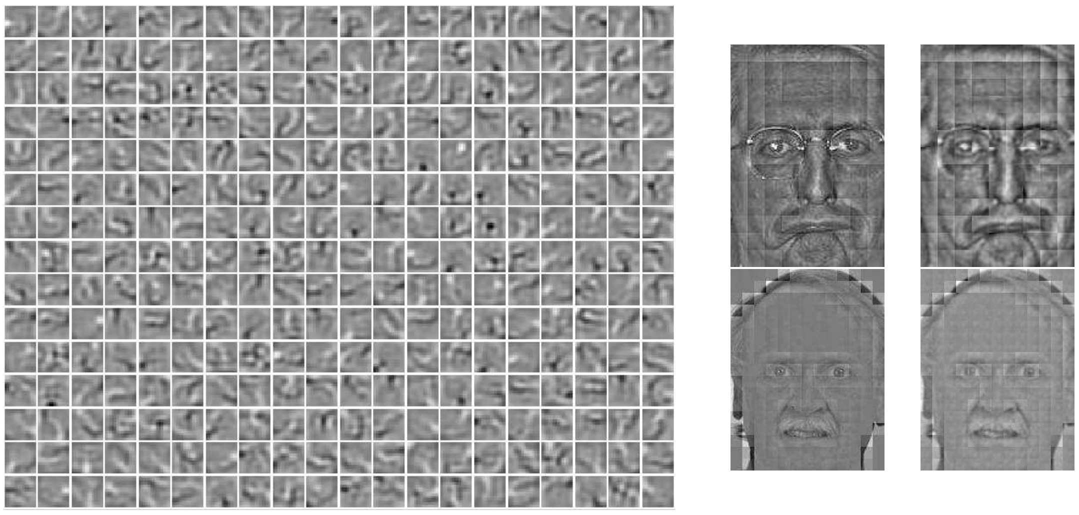


Figure 3.5: Weights of 300 hidden units at 50th iteration and reconstructions

non faces very well, as can be seen from the figure 3.6(see in the reconstruction using rbm50 it is even modeling the small flag posts also). To check further the capacity of

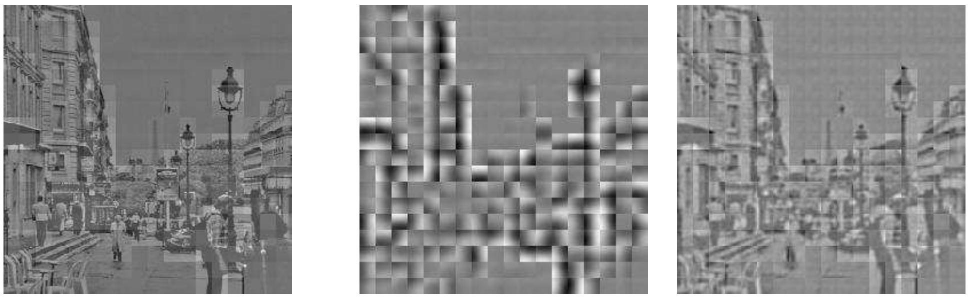


Figure 3.6: Non face input and reconstruction using rbm20 and rbm50 parameters

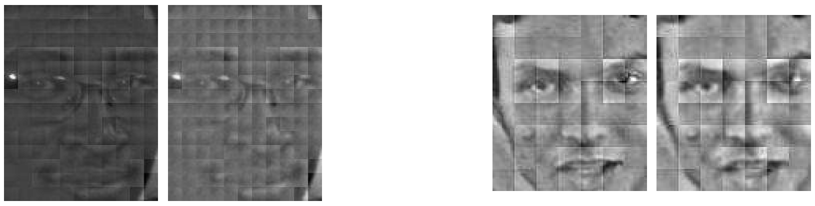
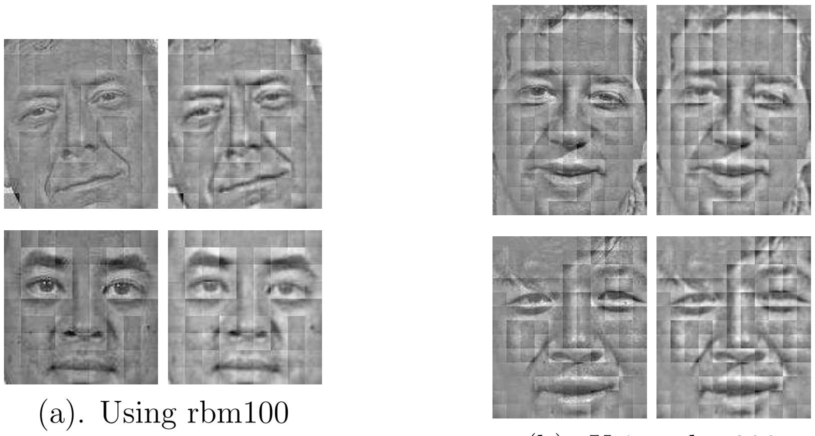


Figure 3.7: Reconstructions of test faces

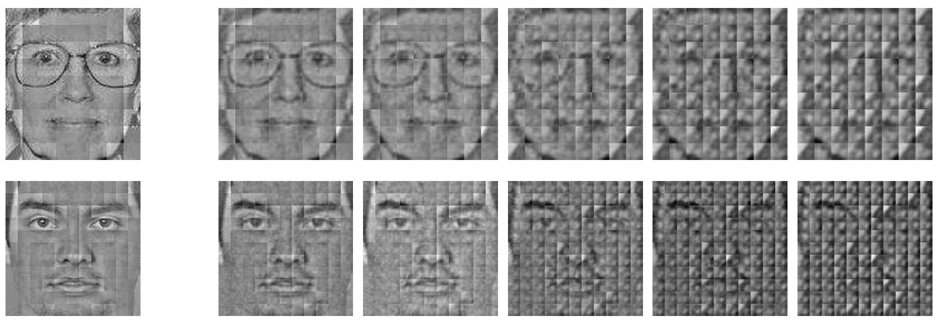

(a). Using rbm100

(b). Using rbm200

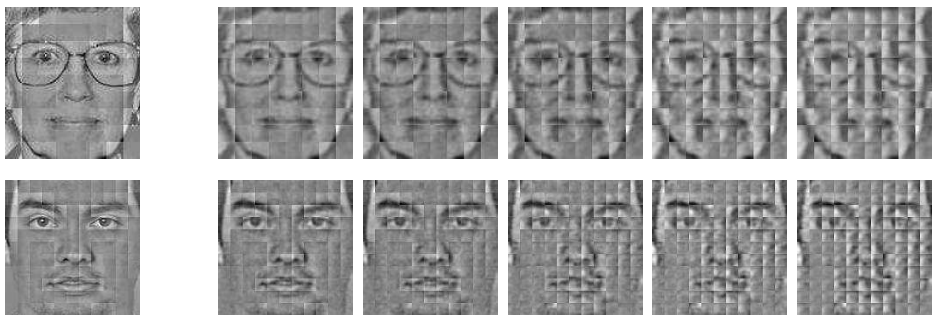
Figure 3.8: Normalized inputs(left) and reconstructions(right)

the model, we tried to run Gibbs cycles for more number of times(10 cycles) and see the reconstruction at each cycle hoping that it will reconstruct faces well when compared to non-faces. The first sample from the cycle didn't differ much for rbm50 and rbm480 as the sampling cycles proceeded after 5^{th} cycle rbm50 is modeling the noise much rather than face features where the rbm480 was doing the good reconstruction see figures 3.9.(a) and (b). Moreover the machine is behaving in a similar manner for both face and non-face reconstructions compare figures 3.9.(b) and (c).

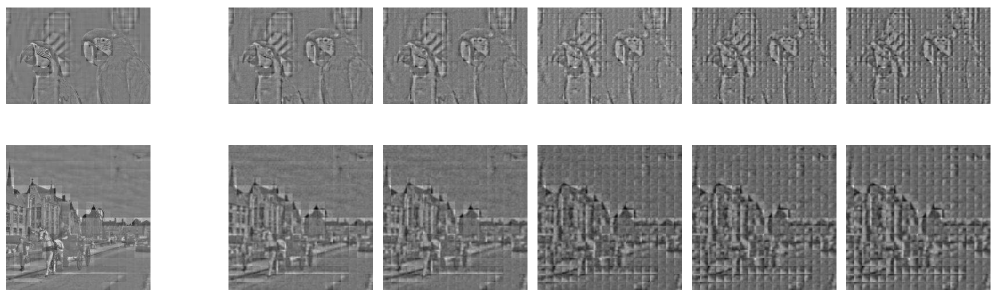
Since the algorithm is behaving similarly for faces and nonfaces we cannot use the



(a)

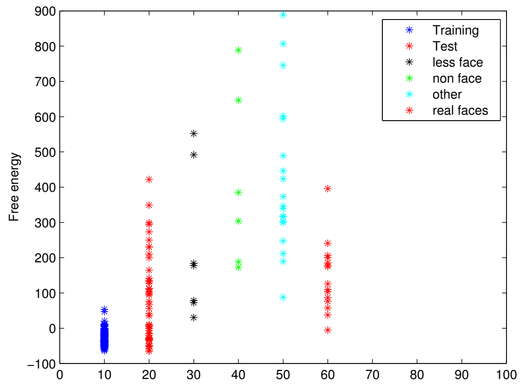


(b)

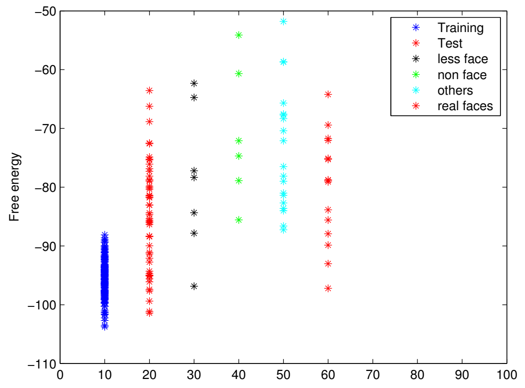


(c)

Figure 3.9: Face Input and reconstructions at 1,2,5,7 and 10^{th} cycles of Gibbs sampling (a)using rbm50 parameters (b)using rbm480 parameters (c)Non-face reconst using rbm480



(a). For normalized input images



(b). For reconstructed images

Figure 3.10: Free Energy differentiation using rbm50

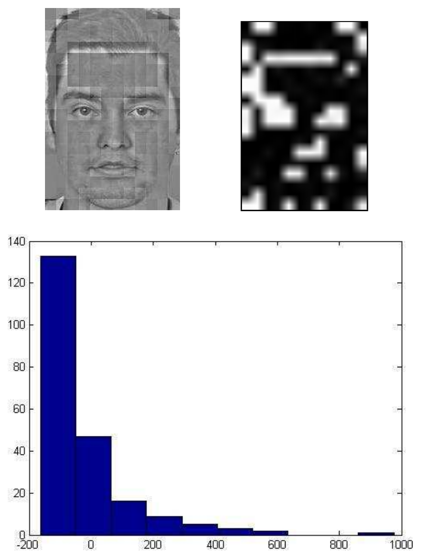
Reconstruction mean square error(MSE) as measure for discrimination. Hence, we used free-energy of an image as a measure for discriminating in the section below.

3.1.3 Free Energy for assessment

Free energy of each patch of an image is calculated and the mean is used as the Free energy of that image, to visualize the free energies of patches of an image we used the imshow[2] command on the free energy matrix. This assessment is done for both the input image(patch normalized) and reconstructed images. Figure 3.10.(a) shows the free energy discrimination plot using rbm50 parameters for patch normalized images, where train faces(blue) are having low energies (-100 to 50) and non-faces(green, black, blue) are having higher energies (greater than 70). However only some of test faces(red) have low energies and others are overlapping significantly with the non-faces this we will see in detail for sample images below. Free energy assessment for reconstructed images also has similar results.

Figure 3.11.(a) shows the input patch normalized image and free energy image along with its histogram and value of mean free-energy where as 3.11.(b) shows the same for the reconstructed image. As we can see from the figures 3.11.(a), 3.12.(a), 3.13.(a) it is giving higher free energies to the patches of high variance i.e eyes, mouth and edges.

[2]assigns black color to min. value and white color to max. value and gray scale in between


(a). Input: -14.941

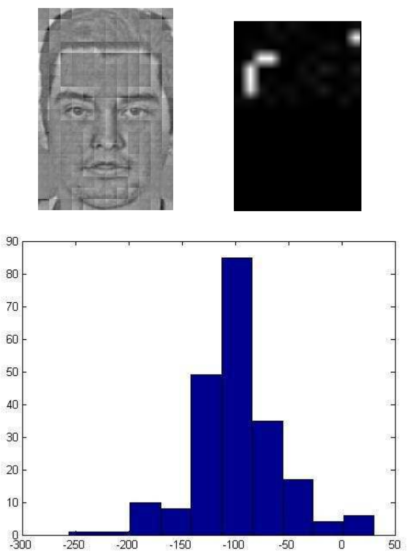
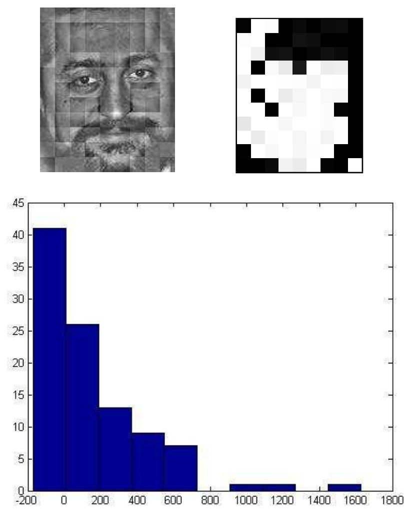

(b). Reconstruction: -98.899

Figure 3.11: Free energy of a Training face and its histogram

For the reconstructed image the variances are reduced (see 3.11.(b), 3.12.(b), 3.13.(b)) and free energies have shifted to the lower levels as can be seen in the histogram plots. In this way, faces will have less patches with high variances when compared to non-faces


(a). Input: 164.58

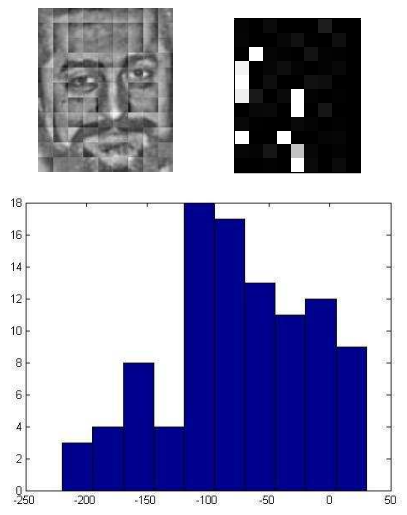
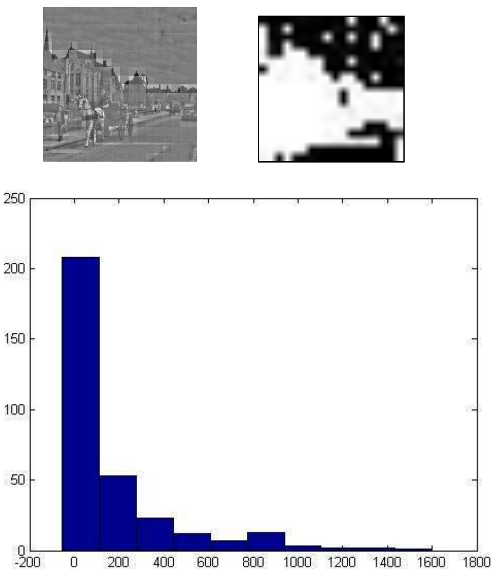

(b). Reconstruction: -75.37

Figure 3.12: Free energy of a Test face and its histogram

resulting in lower free energies whereas non-faces generally have more patches of high variance resulting in higher free energies. However, sometimes things like beard, glasses and some expressions added variance to the faces resulting in the higher free energies


(a). Input: 157.15

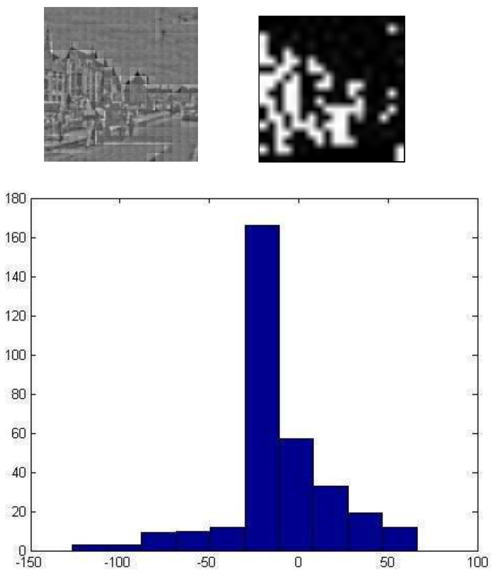
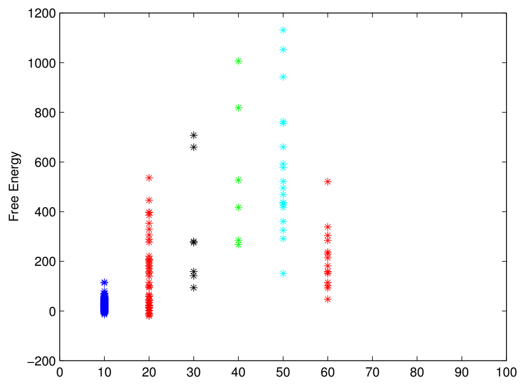
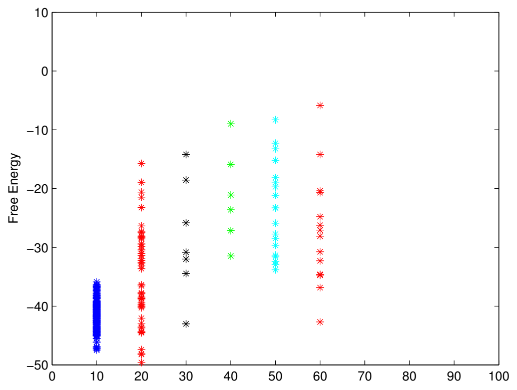

(b). Reconstruction: -12.40

Figure 3.13: Free energy of a Non face and its histogram

causing them to overlap with the non-faces see figure 3.12. Also some smooth natural images sometimes get lower free energies overlapping with the faces.



(a). For normalized input images



(b). For reconstructed images

Figure 3.14: Free Energy differentiation using rbm485

3.2 Whitened Images Training

The dataset used in the earlier section was whitened using the whitening matrix obtained as mentioned in Section 2.3.2 (ZCA whitening). Algorithm parameters used for PCD training of GRBM are given in the below table:

Table 3.2: GRBM Paramters for Whitened data training

Parameter	value
nv	196
nh	1200
ϵ	0.01(Decreased Iterwise)
ϵ_σ	0.0001
weightCost	0.0001
sparseCost	0.001
sparseTarget	0.1(Decreased to 0.05)
Gibbs cycles	1
batchSize	200
maxIter	1000

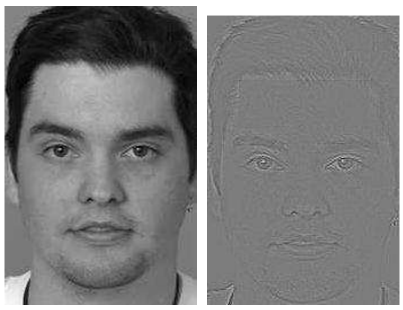


Figure 3.15: Sample input image and whitened image

Training of machine was stopped at 560^{th} iteration when RMSE stopped improving further.

3.2.1 Training Phase

Hidden Units Activation The hidden units activation probabilities at some iterations for a batch are shown below figure 3.16, we can see the sparsity effect as the iterations progress. We can notice vertical lines initially which slowly turn into random activations from that to horizontal lines at the end with intense activations.

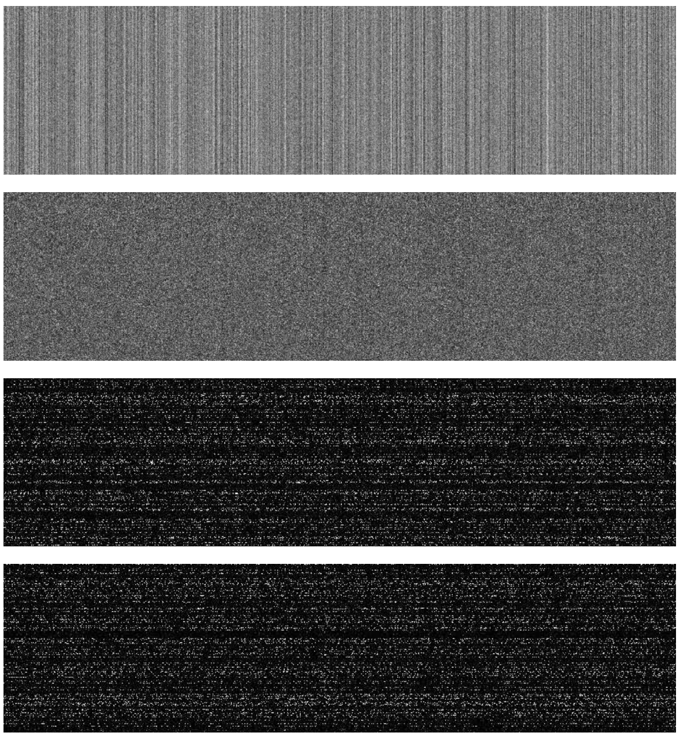


Figure 3.16: Hidden unit activation probabilities for a batch at iter: 1,11,152,466

Error Convergence and Free energy Behavior With whitened data error convergence is smooth (figure 3.17) and its profile has slight variations from the earlier case of patch normalized training. Free energy has increased rapidly till 100 iterations and after that it decreased very slowly (figure 3.18). The amount of overfit (figure 3.18.(b)) is very less till 100 iterations and after that it has increased slightly and remained over there.

3.2.2 Testing and Analysis

Evolution of Weights and Reconstructions Weights in the whitened domain as shown in figures 3.20 and 3.23 can't be interpreted for some meaningful features yet they are very well able to reconstruct the details of face like eyes, hair and expressions. However, the dewhitened weights as shown in figures 3.19 and 3.22 are initially noisy and slowly evolved to smooth textures but didn't seem like capturing any edges. There is no much difference in the reconstructions for 1 gibbs cycle sampling for rbm100 and

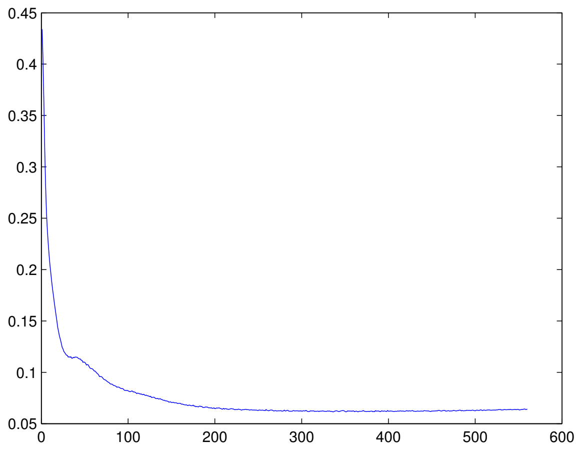
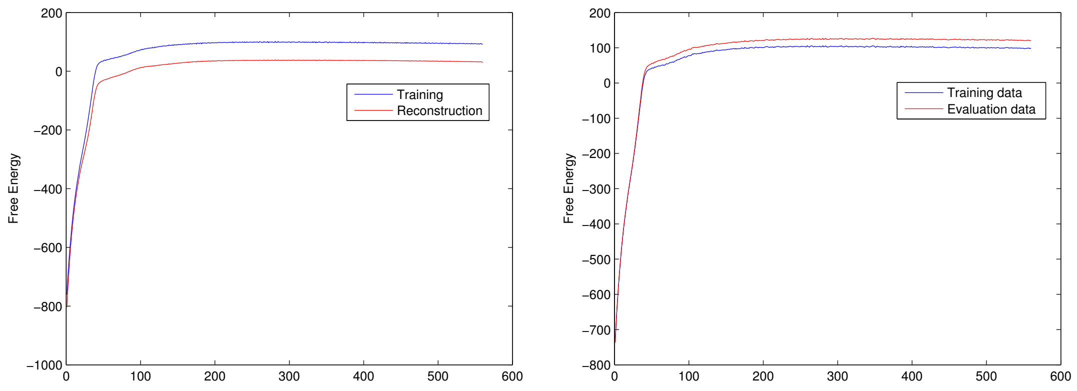


Figure 3.17: Error Vs Iterations



(a). Train and Reconstruction (b). Train and Evaluation

Figure 3.18: Free Energy Vs Iterations

rbm350. For the whitened images training also machine is behaving in a similar manner for both face and non-face images see figure 3.24.

The gibbs sampling was run for 10 cycles and reconstructions are shown in figure 3.25, as the samplings progress after 5^{th} cycle rbm350 is performing better than rbm100. Moreover machine is doing similar reconstructions for face and non-face even for more number of gibbs samplings.

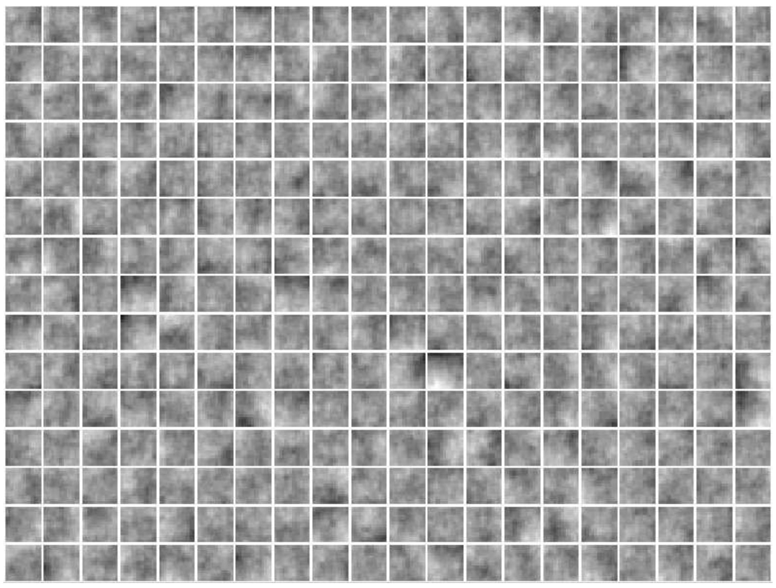


Figure 3.19: Dewhitened weights at 100th iteration

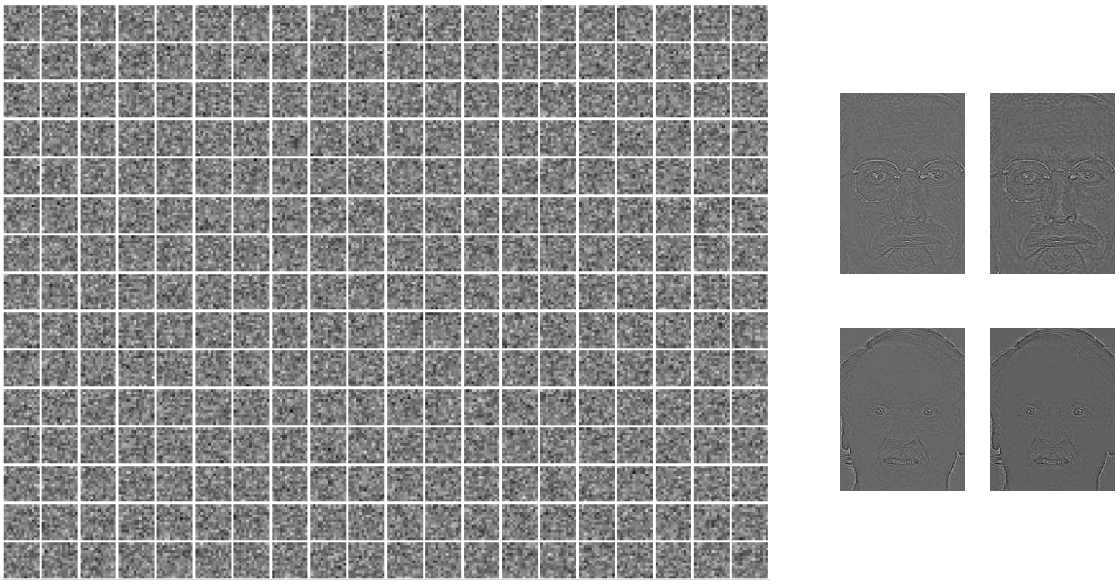


Figure 3.20: Weights and Reconstructions at 100th iteration

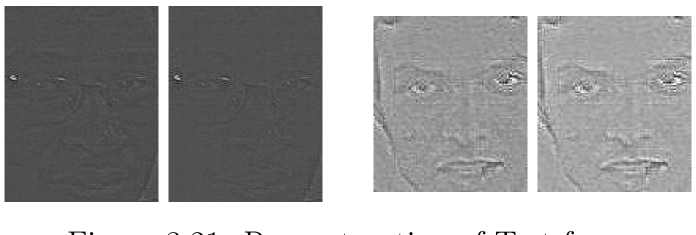


Figure 3.21: Reconstruction of Test faces

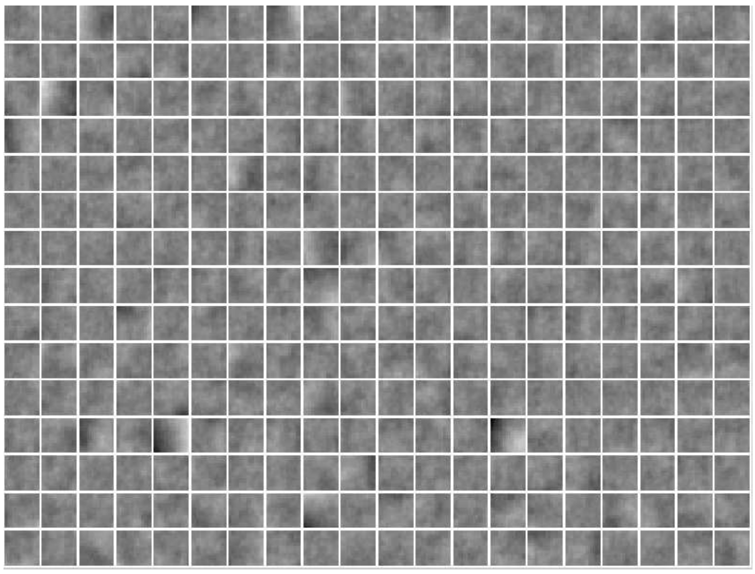


Figure 3.22: Dewhitened weights at 350th iteration

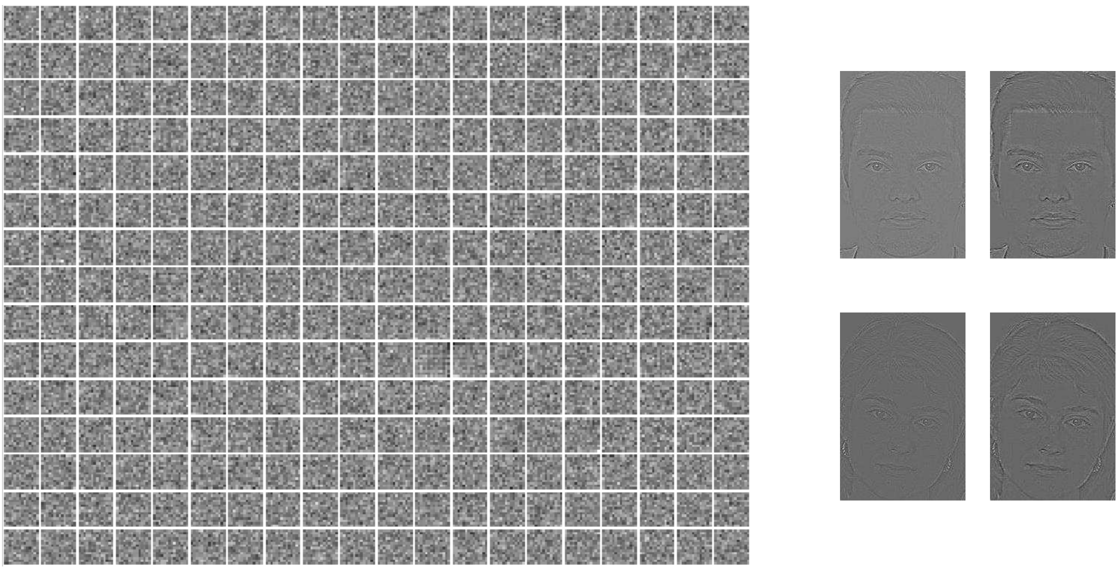
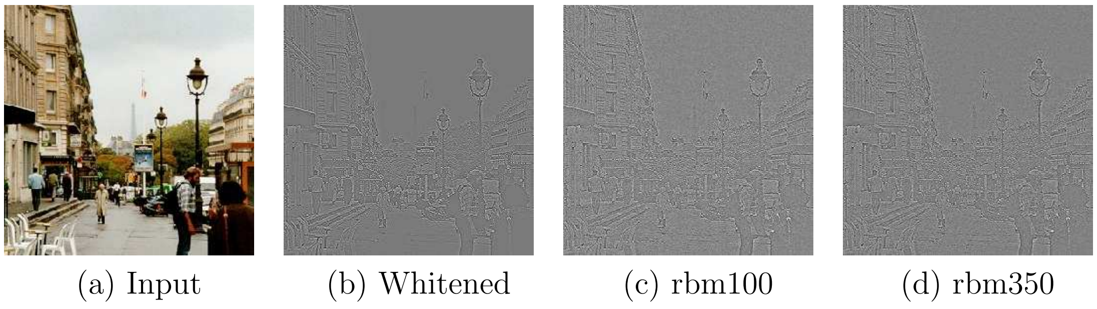
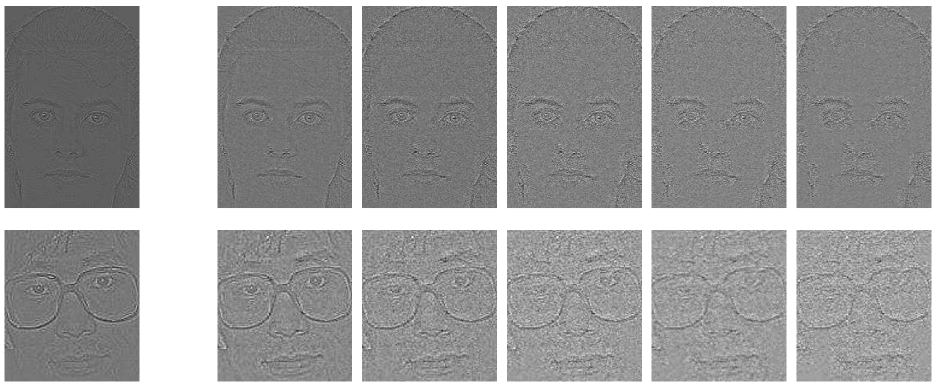


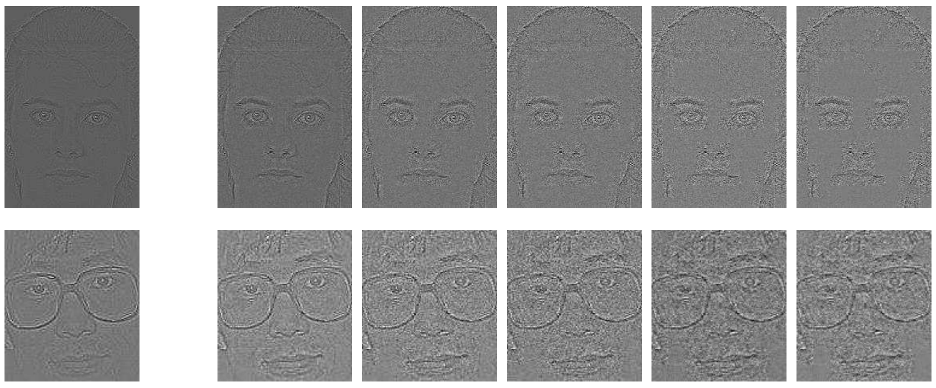
Figure 3.23: Weights and Reconstructions at 350th iteration



(a) Input (b) Whitened (c) rbm100 (d) rbm350

Figure 3.24: Reconstruction of Non-face (a), (b) inputs (c), (d) reconstructions


(a)


(b)

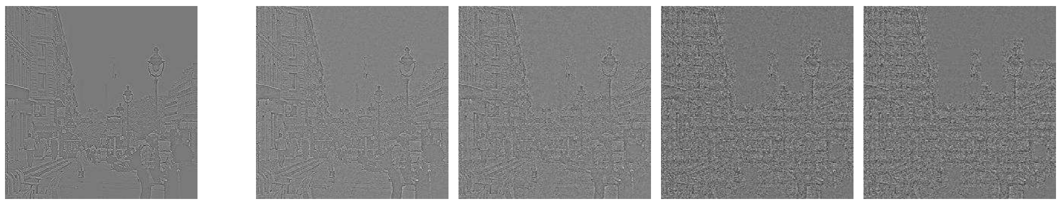
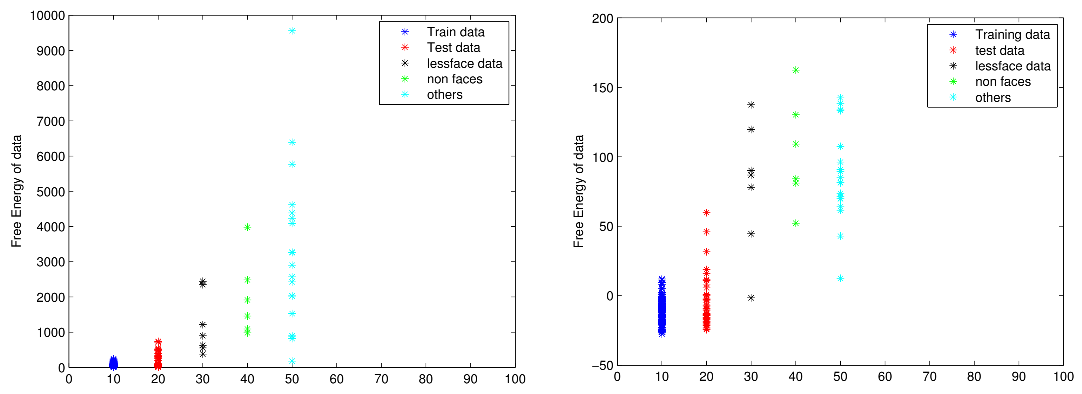

(c)

Figure 3.25: Face Input and reconstructions at 1,3,5,8 and 10^{th} cycles of Gibbs sampling (a)using rbm100 parameters (b)using rbm350 parameters (c)Non-face reconst using rbm350

3.2.3 Free Energy for assessment

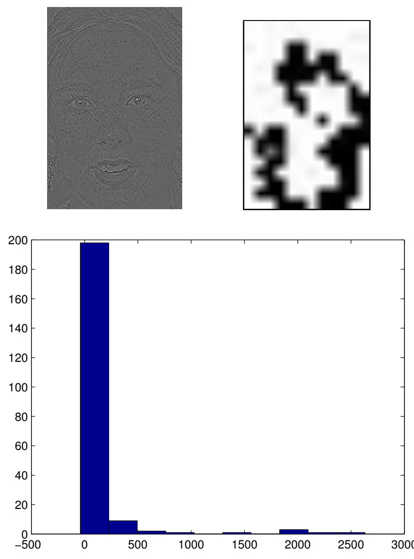
Free energy is used for assessment in the same way as described in earlier Section 3.1.3, here also machine is giving the higher energies to the patches of high variance and lower energies to the lower energies to the patches of low variance. Figure 3.26 shows the free energy assessment for faces (training and test) and non-faces, training faces(blue) are having lower energies (-40 to 20) whereas non-faces are having higher energies (greater than 40). For whitened data training, machine is doing good discrimination for the reconstructed data unlike the earlier case (patch normalized) which was doing well for input data itself. However, for this case also some of the test faces (red) are overlapping with the non-faces.



(a). For normalized input images (b). For reconstructed images

Figure 3.26: Free Energy differentiation using rbm100

Figure 3.27.(a) shows a whitened input and its free energy image along with its histogram and 3.27.(b) shows the same for reconstructed image, we can see that it is giving higher energies to the patches near eye, nose, mouth and edges. Since the machine is capturing facial features well for the reconstructed images we will analyze their free energy and its histogram further. Reconstructed whitened face images have a peak at around -30 (which corresponds to smoother regions) and then they decay continuously with slight variations see figure 3.27.(b), 3.28.(b) whereas for non-faces after this peak at -30 they have Gaussian peak which is clearly visible in figure 3.30.(b) and also noticeable in 3.31.(b). Even for the faces sometimes things like beard, glasses and expressions also


(a). Input: 108.3490

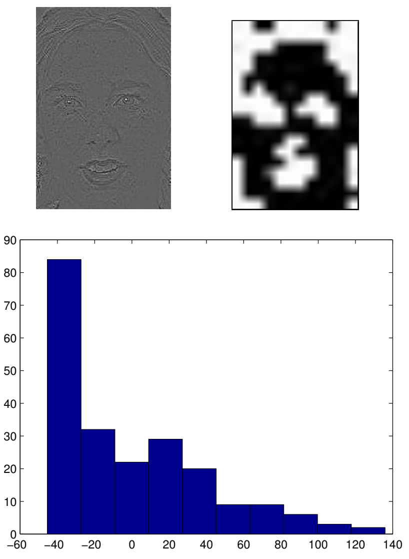
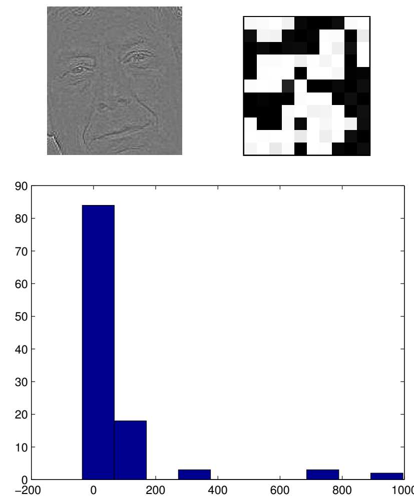

(b). Reconstruction: -1.2831

Figure 3.27: Free energy of a Training face and its histogram

result in this peak in the histogram see figure 3.29.(b).


(a). Input: 62.8438

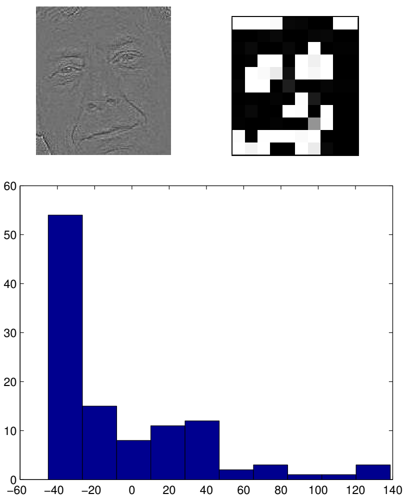
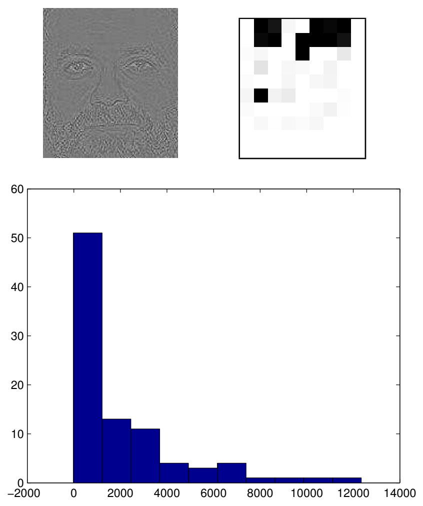
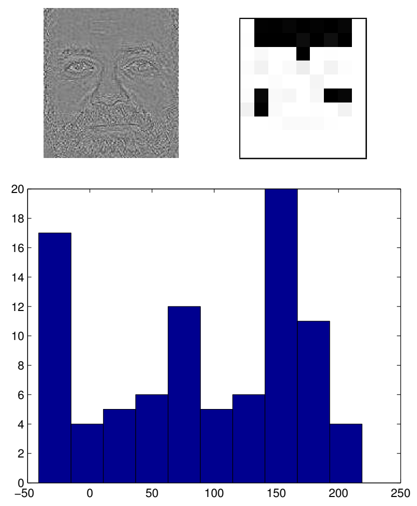

(b). Reconstruction: -7.6038

Figure 3.28: Free energy of a Test face and its histogram

From figure 3.29.(a) and (b) free-energy images we can see that beard region amounting for the higher variances there by to high free-energy(89) than normal faces(-30 to 10) thus causing it to overlap with the non-face. From figure 3.30, we can see that although it is a non-face because of high amount of smooth region(sky) it has got less free-energy overlapping with the faces. Figure 3.32 shows the free-energy differentiation obtained using rbm350 which not very much different from the one obtained using rbm100 shown in figure 3.26.

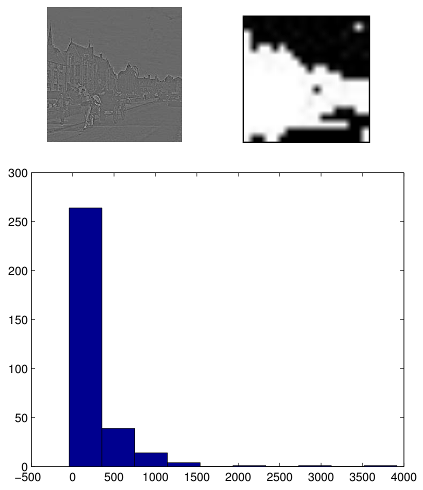


(a). Input: 1860.4

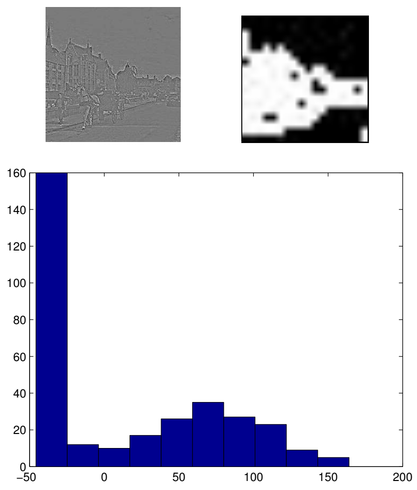


(b). Reconstruction: 89.215

Figure 3.29: Free energy of a Test face and its histogram

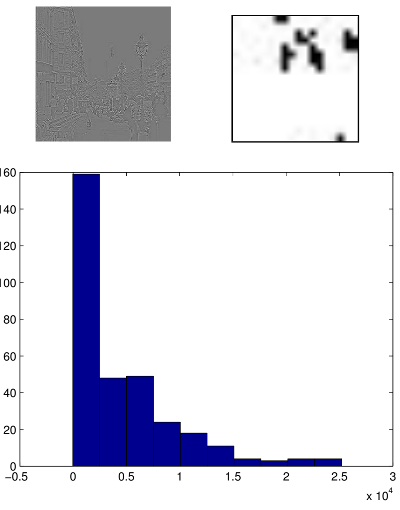


(a). Input: 178.66

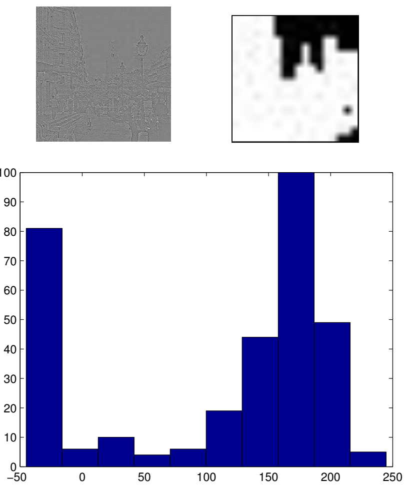


(b). Reconstruction: 12.485

Figure 3.30: Free energy of a non-face and its histogram

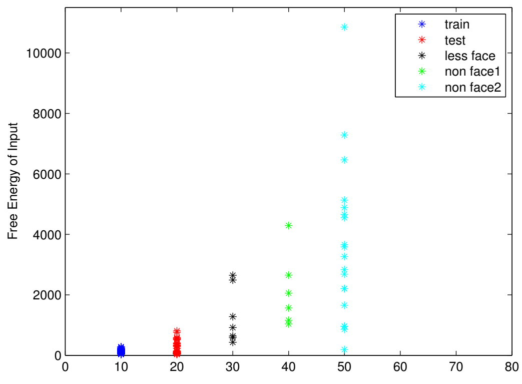


(a). Input: 4386.9

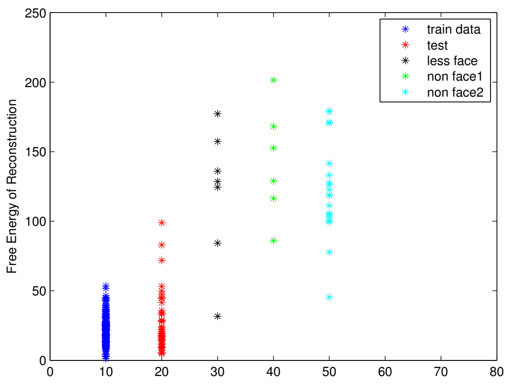


(b). Reconstruction: 107.46

Figure 3.31: Free energy of a non-face and its histogram



(a). For normalized input images



(b). For reconstructed images

Figure 3.32: Free Energy differentiation using rbm350

3.3 Conclusions

Although the machine was trained well, we didn't get the expected result i.e. it was trained only with the patches of faces but it is able to reconstruct even other natural images very well for both the cases of preprocessing. The free-energy discrimination method didn't help much, it is just giving higher energy to the patches of high variance and vice versa. In this way it is giving high free energies even to some good faces just because of beard, glasses and some expressions. Moreover the trivial way of taking the mean of free-energy of all patches didn't take into account facial features captured by the machine as seen in the free-energy images of faces.

We have given the machine great amount of capability with huge number of hidden units to learn from very low patch size (from a high resolution face image) that is why it is able to reconstruct natural images in general. Actually with this training we want the machine to assess face globally by learning the local features of it, but here we have trained only one machine for all the features which has made it learn all sort of filters and hence is able to reconstruct even non-faces quite well.

Chapter 4

Future Work

Since our approach of training one RBM for all the local features didn't work well, we want to resize the cropped face images to 64×64 size and then divide it into overlapping patches of size 16×16 and then train individual RBMs for each patch and again combine them to make a big RBM, this method was proposed by Krizhevsky (2009)[25], here we want to use appropriate number of hidden units to restrict degree of freedom of the machine.

Although machine performed well the traditional energy decreasing didn't happen, so we would like to check the log probability by approximating log-likelihood using Annealed Importance Sampling (AIS) to assess the training of the machine. And also we want to use some DBN architecture instead of simply using free-energy for Face Quality Assessment.

Appendix A

Maths of RBM

Still I have to do this however it is just copy work please refer this [25] for all the derivations, sorry I don't have patience and time to do this.

References

[1] http://www.nist.gov/itl/iad/ig/colorferet.cfm

[2] G. E. Hinton. *Boltzmann machine.* Scholarpedia, 2(5):1668, 2007.

[3] http://www.en.wikipedia.org/wiki/Restricted_Boltzmann_machine

[4] Hinton, G. E. Relaxation and its role in vision. PhD Thesis, University of Edinburgh.

[5] Hinton, Geoffrey E., Terrence J. Sejnowski, and David H. Ackley. *Boltzmann machines: Constraint satisfaction networks that learn.* Pittsburgh, PA: Carnegie-Mellon University, Department of Computer Science, 1984.

[6] Hopfield, John J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79.8 (1982): 2554-2558.

[7] Waltz, David L. Understanding scenes with shadows. (1971).

[8] M. T. Harandi, M. N. Ahmadabadi, and B. N. Araabi. Optimal local basis: A reinforcement learning approach for face recognition. *International Journal of Computer Vision*, 81(2):191204, 2009.

[9] C. Sanderson and B. C. Lovell. Multi-region probabilistic histograms for robust and scalable identity inference. In *Lecture Notes in Computer Science (LNCS)*, volume 5558, pages 199208, 2009.

[10] Y. Wong, C. Sanderson, S. Mau, and B. C. Lovell. Dynamic amelioration of resolution mismatches for local feature based identity inference. In *International Conference on Pattern Recognition (ICPR)*, pages 12001203, 2010.

[11] W. Zhao, R. Chellappa, A. Rosenfeld, and P. Phillips. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399458, 2003.

[12] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar. Attribute and simile classifiers for face verification. In *Int. Conf. Computer Vision (ICCV)*, pages 365372, 2009.

[13] A. Sanin, C. Sanderson, and B. C. Lovell. Improved shadow removal for robust person tracking in surveillance scenarios. In International Conference on Pattern Recognition (ICPR), pages 141144, 2010.

[14] S. Shan, W. Gao, B. Cao, and D. Zhao. Illumination normalization for robust face recognition against varying lighting conditions. In *Workshop on Analysis and Modeling of Faces and Gestures (AMFG)*, pages 157164, 2003.

[15] A. Hadid and M. Pietikäinen. From still image to video based face recognition: An experimental analysis. In Proc. Automatic Face and Gesture Recognition (AFGR), pages 813818, 2004.

[16] S.-A. Berrani and C. Garcia. Enhancing face recognition from video sequences using robust statistics. In IEEE *International Conference on Video and Signal Based Surveillance (AVSS)*, pages 324329, 2005.

[17] X. Gao, S. Z. Li, R. Liu, and P. Zhang. Standardization of face image sample quality. In *ICB, Lecture Notes in Computer Science (LNCS)*, volume 4642, pages 242251, 2007.

[18] K. Nasrollahi and T. B. Moeslund. Face quality assessment system in video sequences. In *BIOID, Lecture Notes in Computer Science (LNCS)*, volume 5372, pages 1018, 2008.

[19] M. Subasic, S. Loncaric, T. Petkovic, H. Bogunovic, and V. Krivec. Face image validation system. In *International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 3033, 2005.

[20] Hinton, G. E., Salakhutdinov, R. R., Reducing the dimensionality of data with neural networks, 2006.

[21] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., Greedy Layer-Wise Training of Deep Networks, 2006.

[22] Wong, Yongkang, Shaokang Chen, Sandra Mau, Conrad Sanderson, and Brian C. Lovell. Patch-based probabilistic image quality assessment for face selection and improved video-based face recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE Computer Society Conference on, pp. 74-81. IEEE, 2011.

[23] Tieleman, Tijmen. Training restricted Boltzmann machines using approximations to the likelihood gradient. *Proceedings of the 25th international conference on Machine learning.* ACM, 2008.

[24] Hinton G. E., Training Products of Experts by Minimizing Contrastive Divergence, 2002.

[25] Krizhevsky, Alex, and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep* (2009).