# 10

# Hierarchical Bayesian Image Models

Daniel Oberhoff
*Fraunhofer FIT-LIFE*
*Germany*



## 1. Introduction

Despite many years of research object recognition remains a hard task for automated systems. In contrast, the tasks in object recognition come very easily to us humans. This has inspired much work in trying to replicate the human abilities by modeling one or more aspects of the human visual system based on the vast amount of research that has been carried out on it and related systems (i.e. visual systems of animals that share much of the brain structure with us, like cats, mice, and monkeys). Bayesian statistics has proven to be a very powerful too in the design, control, and use of such systems. Also it provides a strong mathematical toolset to combine the vast amount of research that has been carried out in the field of image analysis in particular and pattern recognition in general. In this chapter I will introduce Bayesian image models that are constructed in a hierarchical fashion strongly motivated by the human visual system. To do this I briefly review both Bayesian statistics and relevant neurophysiological and psychophysical findings on the human visual system. Then I proceed to introduce the principle of a Bayesian hierarchical image model of such a system. I then proceed to explain the construction of such systems for various applications and highlight the power but also the problems that these systems possess. In the course of this we will demonstrate the capabilities of some of those systems on artificial and real world tasks and wrap up with a conclusion.

## 2. Generative Bayesian image modeling

The underlying inspiration behind Bayesian image models is to understand the nature of the images. This puts them into the class of *generative* Bayesian models contrasing them with *discriminative* models: A generative seeks to explain the image, usually by introducing some *lattent* or *hidden* variables. A discriminative model on the other side seeks to explain only the dependency of one or more output variables on the image, in the case of object recognition the class and possibly the location of the object. While it seems natural to choose a discriminative model when seeking for an object recognition tool, there are many reasons against this. The basic problem is, that the dependency between data and labels is usually very complex. Consequently the model has to be very complex to capture this dependency. With such a complex model then it becomes hard to even find the right region of the solution space. The reason why this is so hard is because without understanding the structure of the images it is hard to infer anything from them. And understanding the structure of the images is exactly what a discriminative cost function does not reward [1]. Consequently our model should seek

[1] It does of course reward it indirectly if it helps the discriminative task, but in that case it does so very indirectly, making learning algorithms that optimize the cost function in small steps very ineffective.

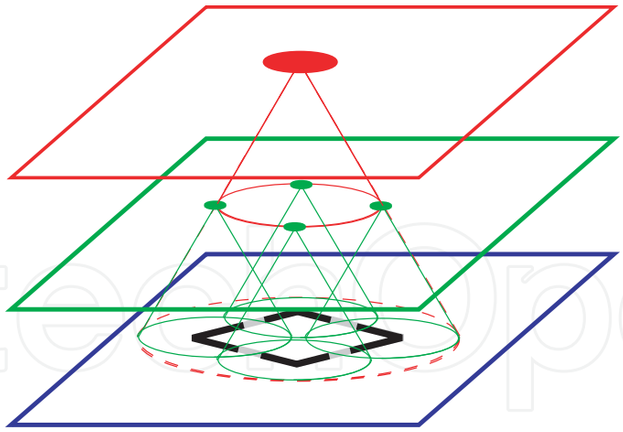


Fig. 1. Illustration of the structure of the hierarchical image model.

to understand the structure of the images first. This is then exactly what a generative model is made for: It tries to understand how the images are made, i.e. how they are *generated*. Since every image is different a generative model will use some internal representation of the image that describes it, but in a more understandable and compact way. This is so because it encodes a probability distribution over all possible images. Since out of all possible pixel configurations, assuming our image representation is pixels, only a relatively small subset actually appears. Knowing this distribution will allow it to be encoded much more compactly than by the pixels directly. This internal representation should then be a much better input to a classifier than the pixels themselves. Going one step further one may also use the complete generative model as a discriminative one, since the composition of a generative model and a classifier is effectively just another, more complex classifier. But starting with a generative model allows the use of learning algorithms that learn to explain the structure of the images. After having done that one can switch to the discriminative cost function to do a *local* search in parameter space to optimize the discriminative performance.

In the following I take an understanding of Bayesian probability formulations as a prerequisite, and also make extensive use of *graphical models* to formulate and depict the various model components. A detailed explanation of these techniques can be found, for example, in Bishop (2006). I also make sporadic use of an extension to the classical graphical model formulation called *gates* which have been ontroduced recently by Tom Minka and John Winn Minka & Winn (2008) and are very useful in the context of models wich are conditioned on discrete latent variables as is the case with the models I introduce in the following sections.

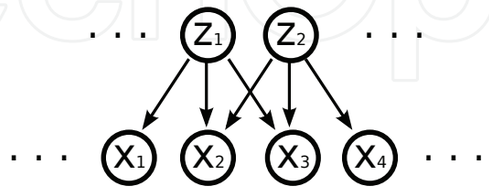
### 2.1 The hierarchical image model

The hierarchical image model is inspired by observations of the function of the human brain and the brain of animals such as cats and monkeys, which are very similar in regard to low- and mid-level vision processing. An important finding in this respect is the fact that visual processing takes place in a step-by-step fashion: The visual information travels through a series of localized cortical areas before it leads to any form of higher level cognition or motor action. The neurons in these areas which encode and process the information show increasingly complex response patterns to visual stimuli, and also become responsive to larger and larger regions of the observed images; one says they have increasingly large *receptive fields* (see, for example, Chalupa & Werner (2003) for more detailed reviews).

The most straight-forward interpretation of this is that what the brain does is break down

the difficult task of analyzing the visual information into a series of much smaller tasks in the spirit of *the divide and conquer* paradigm. Another important finding is, that neurons that respond to stimuli in neighboring regions of the perceived image will also be neighbors in the respective cortical areas. These two findings then lead to a topology of information processing roughly sketched in figure 1.
Going from this abstraction to a Bayesian model the activations and the input become random variables and the connections between layers are modeled using probability distributions.

### 2.1.1 Categorical layer



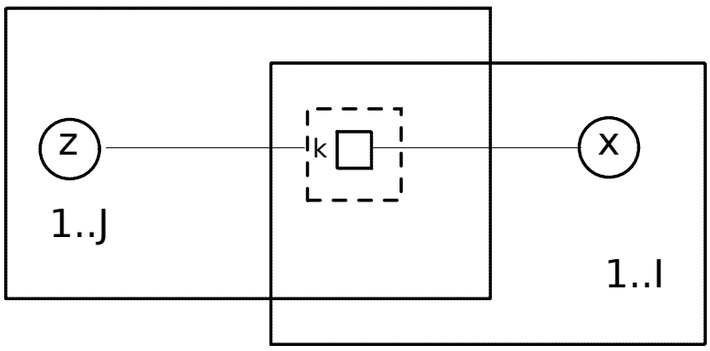




Fig. 2. The categorical model where each component of the data is modeled by one or more parents which are categorical variables selecting a different factor for each setting, parameterized by seperate parameter sets. The predictive distribution is the product of the predictive distributions of all parents.

The central operation of the hierarchical image model is the convergent encoding of variables on one layer into variables on the next layer. In this chapter we limit this choice to the *naive Bayes mixture model*: A mixture model models the probability distribution of an observed variable (which may be of any dimensionality) conditioned on a categorical latent variable $\mathbf{z}$ which can take one out of $K$ values. Thus effectively there is a separate probability distribution over the data for each state of $\mathbf{z}$, each with the same functional form, but with different parameters. Adopting a vectorial notation for $\mathbf{z}$ with only one non-zero entry which is one (a *one-out-of-k-vector*) , the functional form of the mixture model becomes:

$$p(x \mid \mathbf{z}) = \prod_k p(x \mid \Theta_k)^{z_k} \tag{2.1}$$

where $\Theta_k$ are the parameters for each state of $\mathbf{z}$. Note that if the $p(x \mid \Theta_k)$ are from the exponential family then so is $p(x \mid \mathbf{z})$, since its logarithm is the sum of the logarithms of $p(x \mid \Theta_k)$ , multiplied by the corresponding components of $\mathbf{z}$. The functional forms of $p(x \mid \Theta_k)$ I

use in the applications are the *normal* or *Gaussian* distribution over unbounded real continuous variables:

$$p(x \mid \mu, \tau) = \frac{\tau}{\sqrt{2\pi}} e^{\tau(x-\mu)^2} \tag{2.2}$$

and the *multinomial* distribution over categorical variables (in one-out-of-k vector notation):

$$p(\mathbf{x} \mid \text{ß}) = \prod_k \pi_k^{x_k} \tag{2.3}$$

both of which are from the exponential family. I construct multivariate versions of these distributions by taking the product of univariate distributions of the same form but with separate parameters. Such a multivariate distributions built from the product of univariate distributions is called a *naive Bayes* model. The term *naive* refers to the implicit assumption that the input variables are conditionally independent given the latent variable and thus the covariances between input variables vanish. While this indeed is a naive assumption it simplifies inference considerably, and has been used with great success in many applications. Given a mixture model the distribution over the latent variable given a data instance is (using Bayes' law):

$$p(\mathbf{z} \mid x) = \frac{p(x \mid \Theta_k)}{\sum_k p(x \mid \Theta_k)} \tag{2.4}$$

where the denominator is the probability of the data $p(x)$ given the model. Thus using a mixture model and Bayes law the data can be encoded as a multinomial distribution by comparing the likelihood of several data models, which is a nice example how complex models can be constructed from simpler ones.

Out of these receptive field models we build a layer of the hierarchical model by arranging the receptive fields on a regular grid such that they cover the input space. The lowest level input space are feature vectors from low-level image processing, arranged in a two dimensional grid corresponding to the feature position in the image, or in the simplest case the image itself or some local image features such as gabor energy or optical flow. The latent variables of the models encoding each receptive field are then arranged in the same fashion as the receptive fields themselves, resulting in a two dimensional map of random variables. This map can then be used as feature map for the next level. In the simplest case neighboring receptive fields to not overlap, such that each input feature is modeled by a single model. If however neighboring receptive fields do overlap it has to be decided how the models are combined to model the individual input features. The simplest way to do this is to take the product of the predictive distributions of the individual models, corresponding to the situation in figure 2. Note that this leads to the *explaining away* effect Wellman & Henrion (1993), since each variable has multiple explanatory causes, which become correlated when the variable is observed. When applying this model I choose to ignore this additional correlation, and let each parent model the receptive field assigned to it independently. This could also be interpreted as replicating the receptive fields for each parent, and modeling each version independently.

In the next section I discuss a modification which offers an alternative to this rather crude treatment that avoids multiple explanations of the same variable altogether.

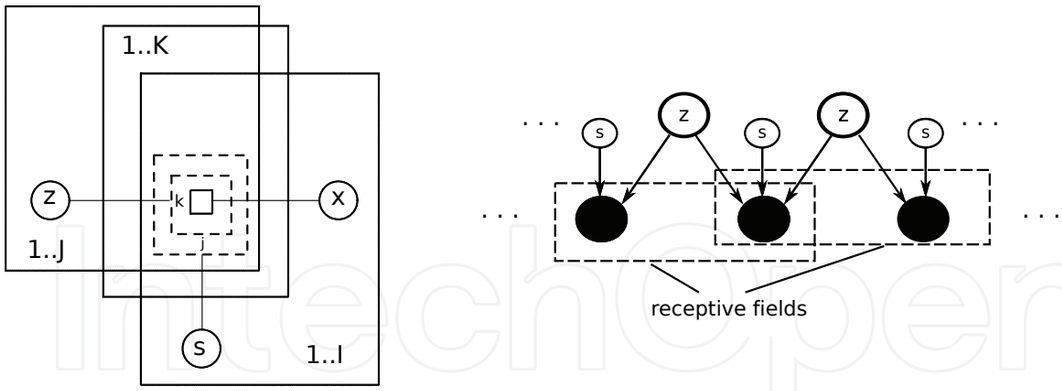



Fig. 3. The factor graph of the gated categorical layer model in gate/plate-notation *gated* left and in the form of a directed graph *right*. The switch variables $s$ select a single parent per datapoint. This effectively induces a segmentation and avoids explaining away effects since no two parents have to explain the same datapoint.

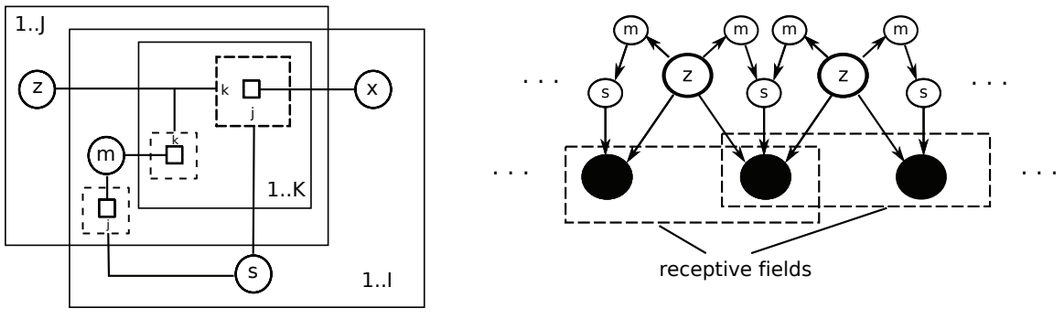



Fig. 4. The factor graph of the gated categorical layer model with additional dependencies between the latent variables and the gates (mediated via the maskbits $m$) in gate/plate-notation *gated* left and in the form of a directed graph *right*.

### 2.1.2 Gated categorical layer

To avoid explaining away I introduce additional auxiliary "switch" variables $s$ for each input feature. The state of these variable then selects which of the models converging on this feature is responsible for explaining it. This setup is shown in figure 3. The joint distribution for a layer with gated mixture models is:

$$p(x,z,s,\Theta) = \prod_{ijk} \left[ p(x_i \mid \Theta_{ijk})^{s_{ij} z_{jk}} \right] p(\Theta) p(s) p(z) \tag{2.5}$$

Besides eliminating explaining away this modification also allows a more natural modeling of occlusion of objects in a complex scene, since an object border (where the background becomes visible) can be modeled directly by a border in the field of the switch variables. This allows the model to model different objects by different internal clusters, instead of having to deal with a multitude of foreground-background combinations within receptive fields.
The switch variables can be embedded in an MRF where the pairwise potentials are chosen to increase the probability that two neighboring gates assign the corresponding data points to the same parent. This increases the probability that contiguous regions of input features are assigned to the same receptive field. If I view this setting as modulating the shape of the receptive fields the MRF can be seen as increasing the likelihood that receptive fields stay

contiguous and don't fall apart into disconnected pieces. To guide the choice of the coupling factor I look towards the two-dimensional Ising Model in physics, which in probabilistic terms is a two-dimensional MRF of binary variables with positive fixed coupling between nearest neighbors. In this model a phase transition between a totally random (corresponding to no effective influence of the MRF on the gate settings) and a totally ordered (corresponding to a completely dominating influence of the MRF) state occurs at a coupling factor of around 1.56 ($e$ to the inverse of the critical temperature as derived by Lars Onsager in 1944 Onsager (1944)). Note that this result had been derived for a four-way connected lattice and the fact that the coupled bits are part of multinomial distributions may further change this value, still Onsager's result is helpful as a rough guide.

A problem with the gated model is that information about object contours is not explicitly represented in the model. This means there is no distinction made between wether an object ends and thus the background becomes visible, or it is occluded by another object, as both induce the same kind of borders in the gate field. This can be fixed by making the shape of the receptive fields variable, thus introducing a dependency between the latent variables and the gates. I mediate this dependency via a mask bit $m_i j$ for each connection between a latent variable $z_j$ and an input $x_i$: If this bit is off the latent variable is effectively disconnected from the corresponding input, meaning it will not try to generate it, and thus is not available for competition for the gate variable. The receptive field shape is encoded in the dependency between the state of the latent variable and a mask bit. This dependency is naturally parametrized by a binomial distribution over the mask bit conditioned on the state of the latent variable. The graphical model for this is shown in figure 4. The joint probability of the model becomes:

$$p(x,z,s,\Theta) = \prod_{ijk}\left[p(x_i \mid \Theta_{ijk})^{s_{ij}}\mu_{ijk}^{m_{ij}}(1-\mu_{ijk})^{1-m_{ij}}\right]^{z_{jk}}\prod_{ij}\left(\frac{m_{ij}}{\sum_{j'} m_{ij'}}\right)^{s_{ij}} p(\Theta) \tag{2.6}$$

## 3. Related work

The attempt to model the neural structure of mammalian brains for image processing approaches has been reflected inseveral schemes for learning and recognition of image patterns. Probably the first such network, called"Neocognitron", was suggested by Kunihiko Fukushima in 1980 Fukushima (1980). Neocognitron consists of a series ofS- and C-layers (mimicking simple and complex cell types, respectively) with shared weights for a set of local receptivefields and inhibitory and excitatory sub-populations of units with interactions resembling neural mechanisms.Neocognitron learns through a combination of winner-take-all competition and reinforcement learning, autonomously formsclasses for presented characters, and correctly classifies some slightly distorted and noisy versions of thesecharacters. In 1989 Yan LeCun et al. LeCun et al. (1989), introduced a similar but much more powerful network for writtencharacter recognition that generated local feature descriptors through back-propagation. A later version of thisnetwork, now called "LeNet", has been shown to act as an efficient framework for nonlinear-dimensionality reduction ofimage-sets Hadsell et al. (2006). "LeNet" is similar in architecture to Neocognitron, but does not learn autonomously andrequires labels to initiate the back-propagation. The latter is not biologically justified and computationallyexpensive.

In 2003 Riesenhuber and Poggio Serre et al. (2005) suggested a computational model for object recognition in the visualcortex with a similar layout called "hmax", in which they put emphasis on the correspondence between model componentsand cortical areas.

"hmax" employs Gaussian radial basis functions to model the selectivity of simple cells, and anonlinear max-function, pooling input from a local population of simple cells, to model functionality of complex cells. Learning in "hmax" is constrained to the tuning of simple cells to random snapshots of local input activity whilepresenting objects of interest. Despite of these simplifications, "hmax" and optimizations thereof where successfullyapplied to the modeling of V4 and IT neuron responses and also used as an input stage to a classifier for object andface recognition yielding very good performance Mutch & Lowe (2006); Serre et al. (2005).

Another approach that focuses very much on the neural details of neural adaptation and learning and does not use weightsharing is found in "VisNet", presented by Deco and Rolls in 2002 Rolls & Deco (2002). The most interesting ingredient totheir model is the fact that it can learn the shift invariance of the feature detectors autonomously through a temporallearning rule called the *trace rule*.

The system we present here is essentially an extension of "hmax", but with a deeper hierarchy and an unsupervisedlearning strategy employed on multiple levels of this hierarchy. This strategy, based on a biologically inspiredcombination of competition and Hebbian update, learns small but informative codebooks after as little as 2 presentationsof the training views. This is in contrast to other codebook optimization approaches which perform gradient descend inan error functionWersing & Korner (2003), which require many steps involving the whole training set. While we put a strongemphasis on a biologically plausible architecture and learning rule, we do not choose to model cortical dynamics indetail. Instead, we utilize biological concepts which help creating a system that performs competitively in terms ofrecognition performance as well as computational load.

The group of T.L. Dean et. al. Dean (2006) have done some groundwork on hierarchical graphical models for computer vision. Theirwork provides great inspiration and technical background, but it has not to our knowledge been applied to real videodata, nor have they tried to do sequence learning.

The most advanced model comparable to the one presented here, has been constructed in the group of Hinton et. al. Hinton (2007).The drawback of their approach is that they use gradient descent learning in combination with sampling methods, andtypically require on the order of thousands of iterations over the data to converge, which seems prohibitive for largevideo sequences.

## 4. Estimation of the hidden variables

Given an image model we need a way to perform some form of algorithmic inference to find out about the states of the hidden variables and finally the posterior distribution of the presence of an object in the image. In the following we thus review here the two most important algorithms for performing inference in a graphical Bayesian model.

### 4.1 Belief propagation

*Belief propagation* is an algorithm that is based on the work of Judea Pearl in the early eighties Pearl (1982). It has the nice property that its computational cost is linear in the size of the model while actually being exact in the case of tree shaped model graphs, where it is effectively an application of the *dynamic programming* paradigm Eddy (2004) to the marginalization problem stated in equation **??**: Consider the factor graphical model in figure 5, *left*. In order to determine the marginal distribution over $x$ we need to marginalize out all

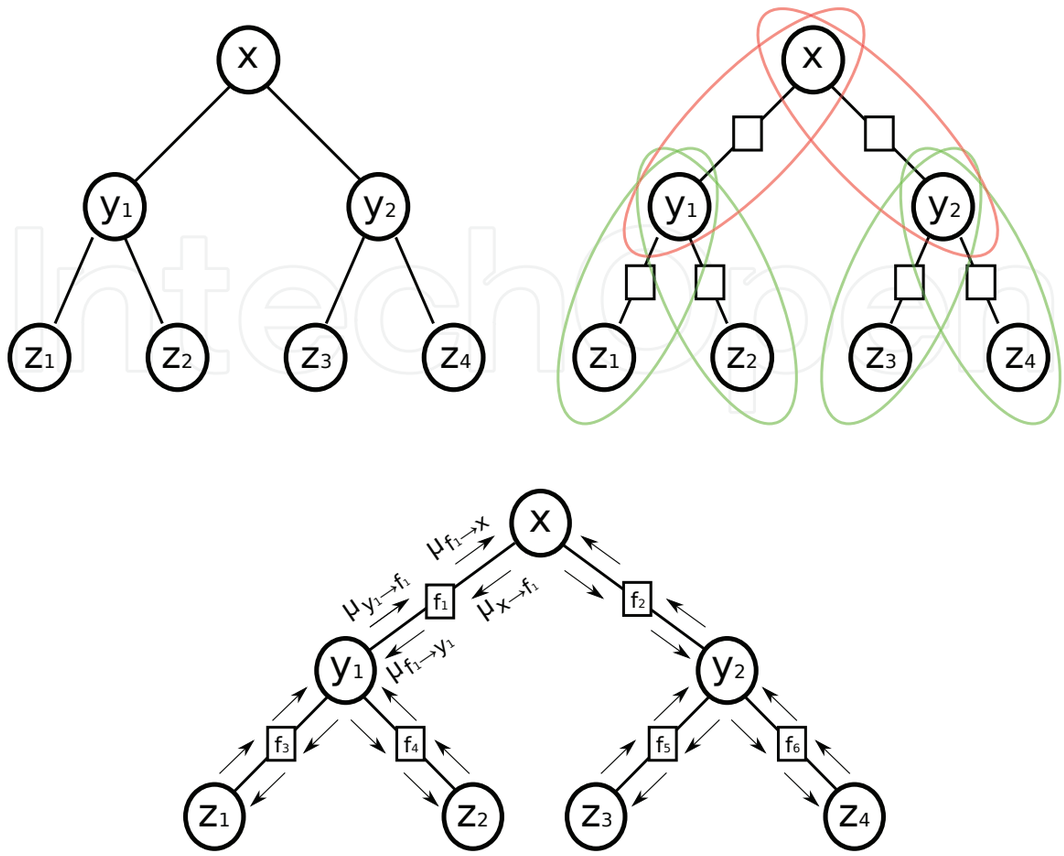



Fig. 5. The undirected dependency graph (*left*) and the corresponding factor graph (*right*) of a probabilistic model for which *belief propagation* is exact.

the other variables:

$$p(x) = \sum_{y_1,y_2,z_1,z_2,z_3,z_4} p(x,y_1,y_2,z_1,z_2,z_3,z_4). \tag{4.1}$$

We can do better than that by exploiting the factorization given by the factor graph in figure 5, *right*:

$$p(x) = \sum_{y_1,y_2,z_1,z_2,z_3,z_4} p(x,y_1)p(x,y_2)p(y_1,z_1)p(y_1,z_2)p(y_2,z_3)p(y_2,z_4) \tag{4.2}$$

$$= \left[\sum_{y_1} p(x,y_1)\left(\sum_{z_1} p(y_1,z_1)\right)\left(\sum_{z_2} p(y_1,z_2)\right)\right] \left[\sum_{y_2} p(x,y_2)\left(\sum_{z_3} p(y_2,z_3)\right)\left(\sum_{z_4} p(y_2,z_4)\right)\right] \tag{4.3}$$

where equation 4.3 is simply the result of rearranging terms by exploiting the associativity of the real numbers. One can see that the sum decomposes into a hierarchy of sums and products that reflects the structure of the factor graph (to make this more clear I have colored the terms for the lowest level of the tree green and for the upper level red, corresponding to the colored ellipses in figure 5, *right*). I will now quickly review the general algorithm in the form found

in Bishop (2006), chapter 8 where, due to the characteristic alternation of sums and products, it is called the *sum-product algorithm*:
Consider a variable $x$ in a graphical model. In order to find its marginal distribution we need to marginalize its adjacent factors with respect to everything else and then take the product of those marginals:

$$p(x) = \prod_{s \in neighbors(x)} \sum_{X_s} F_s(x, X_s) \tag{4.4}$$

$$= \prod_{s \in neighbors(x)} \mu_{f_s \to x}(x) \tag{4.5}$$

where $F_s(x, X_s)$ denotes the effective factor generated by the whole subtree of the graph connected to x via factor $f_s$ and $X_s$ denotes the set of variables in this subtree. Equation 4.4 implicitly defines the *messages* $\mu_{f_s \to x}(x)$ from the factor $f_s$ to the variable $x$. To complete the algorithm we decompose the effective factor $F_s(x, X_s)$ by moving one step further away from $x$ in the graph:

$$F_s(x, X_s) = f(x, x_1, \ldots, x_M) \prod_{m=1}^{M} G_m(x_m, X_{sm}) \tag{4.6}$$

where the $x_m$ are the variables adjacent to the factor $f_s$ except for out root variable $x$ and the $X_{sm}$ are the remaining variables in the subtree beyond $x_m$. The $G_m(x_m, X_{sm})$ are the contributions by the factors adjacent to $x_m$ except for $f_s$:

$$G_m(x_m, X_{sm}) = \prod_{l \in neighbors(x_m) \setminus f_s} F_l(x_m, X_{ml}). \tag{4.7}$$

Now by substituting this into the definition of $\mu_{f_s \to x}(x)$ we get:

$$\mu_{f_s \to x}(x) = \sum_{X_s} f(x, x_1, \ldots, x_M) \prod_{m=1}^{M} G_m(x_m, X_{sm}) \tag{4.8}$$

$$= \sum_{\{x_1, \ldots, x_m\}} f(x, x_1, \ldots, x_M) \prod_{m=1}^{M} \sum_{X_{sm}} G_m(x_m, X_{sm}) \tag{4.9}$$

$$= \sum_{\{x_1, \ldots, x_m\}} f(x, x_1, \ldots, x_M) \prod_{m=1}^{M} \mu_{x_m \to f_s}(x_m) \tag{4.10}$$

which defines $\mu_{x_m \to f_s}(x_m)$. By substituting equation into this definition we can close the recursion:

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm}) \tag{4.11}$$

$$= \prod_{l \in neighbors(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m) \tag{4.12}$$

where the last step again involves the rearrangement of some terms using associativity. Effectively thus calculating the marginals involves two kinds of messages. All of these messages can then be determined by initializing them with one, and then updating each one in turn. To avoid unnecessary calculations one starts at some arbitrary node, updates messages moving away from this node until all the leaves are reached, and then updates messages going all the way back to the starting node.

### 4.2 Sampling

An alternative way of calculating expectations under a distribution over unknown variables is by sampling methods, since sampling from a distribution and averaging over the samples is often easier than performing the required integrals directly. The advantage of sampling methods is that they can often be applied when the methods discussed above are either infeasible or yield very poor approximations. Their drawback is that often a very large amount of samples is required to gain any reasonable amount of information, and for the samples to become de-correlated form the initial conditions and from each other, which would make them useless, or decrease the actual amount of information they convey about the actual distribution of interest. Sampling can also be useful to understand the distribution the model encodes by analyzing a set of samples generated from it.
In the hierarchical image model connections between variables are local, and thus relatively sparse. Correlations between variables that are not directly connected can be expected to be relatively weak. In such a case Gibbs sampling is an appropriate sampling strategy:

- Pick a variable at random or up to a certain specified order.
- Calculate the distribution of this variable given the states of its neighbors.
- Assign this variable to a random sample from this distribution.

We use this form of sampling for the gated categorical layer because belief propagation becomes infeasible due to the large number of messages that would have to be stored and updated.

## 5. Learning

In order to solve our task we need algorithms to optimize our models for given cost functions. As discussed in section 2 we want to have algorithms both for optimizing the generative model, as well as for optimizing the discrimination performance. In the following I will review one algorithm for the generative learning tasks and one for the discriminative task.

### 5.1 Variational Bayesian expectation maximization

In the following I will review a fully Bayesian method of estimating the model paramters. This method has become known under the name of *Variational Bayesian Expectation Maximization* or (VBEM) Attias (1999) and can be seen as a generalization over more classical expectation maximization approaches such as maximum likelihood (ML) or maximum *a posteriori* Dempster et al. (1977).

#### 5.1.1 Conjugate exponential models

Since I wish to learn the exact form of the model distribution from the data I am concerned with encoding knowledge about the parameters given the data. The most natural approach from a Bayesian standpoint is to encode this knowledge in a parameter distribution. To choose the form of this distribution I further note, that a major motivation for the approach I have chosen is the ability to learn unsupervised. Thus I will be interested in the parameter posterior after seeing the data, which is proportional to the product of the parameter prior and the data likelihood

$$p(\Theta \mid \mathbf{x}) \propto p(\Theta) \prod_i p(x_i \mid \Theta) \tag{5.1}$$

If I further allow continuous updating of the posterior, i.e. using the posterior from the data seen so far as prior for the following data, it becomes desirable that the functional form of the posterior be the same as that of the prior. Given that $p(x \mid \Theta)$ is from the exponential family a prior of the form

$$p(\theta|\eta,\nu) = g(\theta)^{\eta} h(\eta,\nu) e^{\phi(\theta)^T \nu} \tag{5.2}$$

will lead to a posterior of the same form with parameters

$$\eta' = \eta + N \tag{5.3}$$

$$\nu' = \nu + \sum_{n=1}^{N} u(x_n) \tag{5.4}$$

where $N$ is the number of data points. The prior thus encodes a set of previous observations (or pseudo-observations if no data has actually been seen) where $\eta$ plays the role of a counting variable and $\nu$ accumulates the observations sufficient statistics. Such distributions exist for many of the exponential family distributions. They are commonly referred to as the distributions' *conjugate* prior. A comprehensive list of conjugate priors for commonly used distributions can be found in Fink (1995) and a table of the distributions I use in this thesis together with some important properties can be found in the Appendix.
If all parameters of a probabilistic model are equipped with conjugate priors they are called *conjugate exponential* (CE). The probabilistic image models I construct in this thesis are built from conjugate exponential model components and also in composition always remain conjugate exponential.

### 5.1.2 Bayesian posterior approximation

Given a parameter prior and some data we wish to infer the posterior parameter distribution and the marginal data likelihood. This becomes intractable in the presence of hidden variables, as one would have to infer about both simultaneously:

$$p(\Theta, \mathbf{Z}, \mathbf{X}) = p(\Theta, \mathbf{Z} \mid \mathbf{X}) p(\mathbf{X}) \tag{5.5}$$

$$= p(\mathbf{X} \mid \Theta, \mathbf{Z}) p(\mathbf{Z} \mid \Theta) p(\Theta). \tag{5.6}$$

To attack this problem one begins by introducing an approximate distribution $q(\Theta, \mathbf{Z})$ and decomposes the log-marginal probability of the data into a the expectation given the approximation and an approximation error term:

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \mathrm{KL}\left(q \parallel p\right) \tag{5.7}$$

$$\mathcal{L}(q) = \int\int q(\Theta, \mathbf{Z}) \ln \left( \frac{p(\Theta, \mathbf{Z}, \mathbf{X})}{q(\Theta, \mathbf{Z})} \right) d\mathbf{Z} d\Theta \tag{5.8}$$

$$\mathrm{KL}\left(q \parallel p\right) = -\int\int q(\Theta, \mathbf{Z}) \ln \left( \frac{p(\Theta, \mathbf{Z} \mid \mathbf{X})}{q(\Theta, \mathbf{Z})} \right) d\mathbf{Z} d\Theta \tag{5.9}$$

where $\mathcal{L}(q)$ is a lower bound since the error term, given by the Kullbach-Leibler divergence between the approximation and the true posterior, is always positive. Thus I can minimize the error indirectly by maximizing the lower bound. We now turn our attention to approximations $q(\Theta, \mathbf{Z})$ which factorize between the parameters and the hidden variables:

$$q(\Theta, \mathbf{Z}) = q(\Theta) q(\mathbf{Z}). \tag{5.10}$$

In this case the lower bound decomposes into a sum of negative Kullbach-Leibler divergences:

$$\mathcal{L}(q) = -\mathrm{KL}\left(q(\Theta) \parallel p(\Theta, \mathbf{Z}, \mathbf{X})\right) - \mathrm{KL}\left(q(\mathbf{Z}) \parallel p(\Theta, \mathbf{Z}, \mathbf{X})\right) \tag{5.11}$$

and we get an EM-algorithm by optimizing each one in turn while keeping the other one fixed. Note that not further assumptions about the nature of the approximations entered the derivation. For example in one of the experiments I resort to sampling to approximate the hidden variable distribution $q(\mathbf{Z})$, which does not change the validity of the optimization, even though it might affect its quality.
When the parameter posterior approximation $q(\Theta)$ has the form of a Dirac delta function, then the variational optimization yields the maximum likelihood and maximum a posteriori variants described above. The difference between the two then lies in the choice of paramter prior, which in the case of ML is implicitly uniform [2]
In the more interesting case that the prior is conjugate to the model, i.e. the model is conjugate exponential, the true posterior has the same form as the prior. Naturally the approximation is then chosen to also have the same form as the prior. In this case the first KL in equation 5.11 can be reduced to zero by setting:

$$q(\Theta) = e^{\langle \ln p(\Theta, \mathbf{Z}, \mathbf{X}) \rangle_{q(\mathbf{Z})}}. \tag{5.12}$$

Since $q(\Theta)$ is conjugate to $p(\Theta, \mathbf{Z}, \mathbf{X})$ this amounts to collecting the expected sufficient statistics and the prior:

$$p(\Theta) \propto g(\Theta)^{\eta} e^{\phi(\Theta)^T \nu} \tag{5.13}$$

$$q(\Theta) \propto g(\Theta)^{\eta'} e^{\phi'(\Theta)^T \nu'} \tag{5.14}$$

$$\nu' = \nu + \sum_{i=1}^{N} \langle u(\mathbf{Z}, \mathbf{X}) \rangle \tag{5.15}$$

$$\eta' = \eta + N. \tag{5.16}$$

### 5.2 Stochastic gradient descent

To optimize the discriminative performance of the model I turn to direct optimization via gradient descent. Effectively this means that I train a discriminative model that has the same structure as the generative model, and is initialized with the paramters learned by the generative model (see Lasserre et al. (2006) for a detailed discussion of this approach). That is we take the gradient of a cost function that is high when the performance is good and low otherwise, and take a step in the direction of the gradient. If the cost function is sufficiently smooth this will lead us at least to a local optimum once the gradient vanishes and the descent algorithm converges. In deep models like the hierarchical image models one can make use of dynamic programming to vastly reduce the computational effort. To see this consider taking the gradient with respect to a parameter attached to a factor at the far end of the network. Using the chain rule we find this gradient by propagating the gradient with respect to the network output by propagating it through the Jacobians of the factor along every possible path between the output and this factor, and summing up the contributions of each path. Now considering we want the gradient with respect to the parameters of all factors we see

[2] A uniform prior can often not be normalized, which is then known as an *improper prior*. Nevertheless using an improper prior yields a valid posterior, up to normalization of course.

that many of the partial results can be reused. We thus propagate step by step in parallel form the output to the input, and simply store the partial results which are the gradients with respect to the factors along the way.
The proper way of performing gradient descent would be to average the gradient over the whole training set for each step. In the case of image data the amount of data becomes large very quickly. So instead I apply the gradient averaged over one image immediately after the image was presented to the network. This way of using derivatives of part of the training set is referred to as *stochastic gradient descent* (see e.g. LeCun et al. (1998) for a more detailed discussion). Averaging overr images also reduces the dominance of larger images in the gradient, which helps reducing training set bias when images of one class are much larger than images of the other classes.
In the context of this model the classification is performed by a naive Bayes classifier operating on the representation established by the highest layer of the hierarchical model. Discriminative training of this setup can be interpreted as training a non-linear logistic regression classifier with the model itself as the kernel function. As error function I consequently use the cross-entropy error which leads to an error derivative linear in the distance between predicted and actual class probabilities (see e.g. Bishop (1995) for an extended treatment of the appropriate choice of error function).
Since the error landscape will most probably not be very isotropic, that is the gradients will have widely varying magnitudes in different regions of the parameter space, we employ a momentum term that will speed up movements in directions which are consistent in consecutive evaluations, while slowing down directions which are not. While this is a very simple technique and more sophisticated techniques exist, it is easy to implement and presents little numerical difficulty. Also it can readily deal with consistent directions which are diagonal to several dimensions, which is not the case for adaptive learning rate algorithms such as the widely used diagonal Hessian Levenberg-Marquardt algorithm.

### 5.3 Weight sharing

To decrease the number of parameters and to exploit the fact that the presence of patterns in the image is roughly independent of the position in the image I tie parameters of the multiple receptive fields in a layer together. For learning this means that the statistics from the individual receptive fields (or the gradients in gradient descent) are summed together. This also allows the model to adapt to varying image geometry by adjusting the number of receptive fields. This effectively makes the model a kind of a *convolutional model* since the scanning of the image (or a higher layer output) is reminiscent of a convolution operation, or would be if neighboring receptive fields would overlap maximally.

### 5.4 Training set bias

When the training set is strongly biased, i.e. data from one class is much more ubiqutous than data from the other classes, the generative model will assign more detail to this class than to the others, since the ststistics of this class have more wight in the parameter update step of VBEM learning (see section 6.2 for empiric observation of this effect on real data). To avoid this I reweigh the statistics of the classes in the update step such that the all classes are approximately weighed equally.

### 5.5 Transformation invariance

To be robust against shifts and scalings in the training set we can use Bayesian marginalization to estimate the optimal shift and scale for each training image. This is done by introducing an additional selector variable that has one possible setting for every allowed shift and scale:

$$p(x \mid \Theta, \mathbf{s}) = \prod_{i,j,l} p(\mathcal{T}_{ij}(x) \mid \Theta)^{s_{ijl}} \tag{5.17}$$

$$p(x \mid \Theta) = \sum_{i,j,l} p(s_{ijl}) p(\mathcal{T}_{ij}(x) \mid \Theta) \tag{5.18}$$

where $\mathcal{T}_{ijk}$ denotes the image transformation, $i$ and $j$ enumerate the allowed shifts, and $l$ enumerates the allowed scales while now $s$ is a selector variable which is one only for one specific scale and shift. The marginalization over shifts may seem a large computational effort at first, but it can be sped up considerably by using some tricks. If a one layer hierarchy is used then the operation on the image at some point amounts to a convolution:

$$p(x \mid \Theta) = e^{\sum_{i'j'} u(x_{i-i'j-j'}) \phi(\Theta_{ij})} \tag{5.19}$$

where $i'j'$ span the receptive field. This operation can be sped up by utilizing the convolution theorem and a the Fast Fourier Transform (FFT). If a two layer hierarchy is used then we can restrict the shifts to multiples of the displacements of neighboring receptive fields of the lower layer. Then we only have to compute lower layer responses once, and can perform the marginalization on the layers outputs, since all receptive fields have the same parameters. To avoid missing information about the intermediate shifts we can also lay out the lowest level receptive field densely (which again allows evaluation of the lowest layer using the FFT), and perform pooling to reduce the resolution using the probabilistic logical or:

$$p(z_1 or z_2 \ldots or z_n) = 1 - \prod_i (1 - p(z_i)) \tag{5.20}$$

Since this yields distributions over bit-vectors instead of one-out-of-k vectors we then use a multivariate binomial as the emission model for the next layer.

### 5.6 Feature selection

In the detection example it may be beneficial to operate only on a subset of the receptive field. For one this makes it possible to deal with non-rectengular object shapes. It also makes it possible to supress regions of the object model which are not reliable for detection. In effect this is somewath similar to the mask bits in the segmenting graphical model. Only that in this case the parts that are masked out remain masked out and are not modelled by any other part of the model. This contradiscts the generative learning scheme, so that this kind of mask has to be learned discriminatively. I define the detection based on the model likelihood like this:

$$p(object \mid \mathbf{l}) \equiv \frac{1}{1 + e^{-(\mathbf{wl} - w_0)}} \tag{5.21}$$

where $\mathbf{l}$ is the vector of log-likelihoods from the individual receptive field positions, $\mathbf{w}$ is a vector of weights for each position, and $w_0$ is a constant offset. To learn the weight vector $\mathbf{w}$ I parameterize each weight using a logistic function to keep the wieghts in the range $[0, 1]$:

$$w_i \equiv \frac{1}{1 + e^{-v_i}}. \tag{5.22}$$

## 6. Experiments

### 6.1 Occlusion modeling

I demonstrate the ability of the model to separate objects that have a constant shape but occlude each other in various configurations on a toy dataset consisting of an artificially generated RGB image sequence in which three geometric shapes of red green and blue color move randomly over a black background, which occasionally occlude each other, examples are shown in figure 6. The parametric form of the emission models $p(x_i \mid \Theta_{kij})$ is a diagonal Gaussian distribution with a normal-gamma prior. Furthermore I put a truncated stick breaking prior distribution on the latent variables, thus the latent variables together with the conditional distributions over variables in their receptive fields approximate a Gaussian Dirichlet Process Mixture Model (DPMM), a non-parametric model which has the ability to find the needed number of components itself, and does not need random initialization in this kind of training procedure, as would be the case of a symmetric dirichlet prior. To speed up training I update the posterior after each image and decay it by a factor of $e^{-1/\tau}$ with $\tau$ equal to ten times the training batch size, which makes the posterior an average over an exponentially decaying window with an effective time window of ten training epochs. To slow down the training at the beginning and avoid premature sharpening of the posterior on the first few images I initialize the stick breaking and normal-gamma posteriors uniformly with a number of pseudo-counts corresponding to again ten training epochs. The topology is such that neighboring receptive fields overlap maximally, i.e. are only shifted by one pixel relative to each other. The animated figures have a size of 8x8 pixels, and the receptive fields are fifteen pixels square, thus each pixel is connected to 255 latent variables. Since each input appears at each possible position in one of the receptive fields it is contained in the model should be able to train a shift invariant representation of them.
To highlight the effects of the introduction of the gate MRF as well and the mask bits I train three different models:

| model | gate MRF | mask bits |
|---|---|---|
| 1 | - | - |
| 2 | x | - |
| 3 | x | x |

Table 1. Overview over the four model configurations analyzed. A cross means the corresponding feature is active in this configuration, a dash means it is not active.

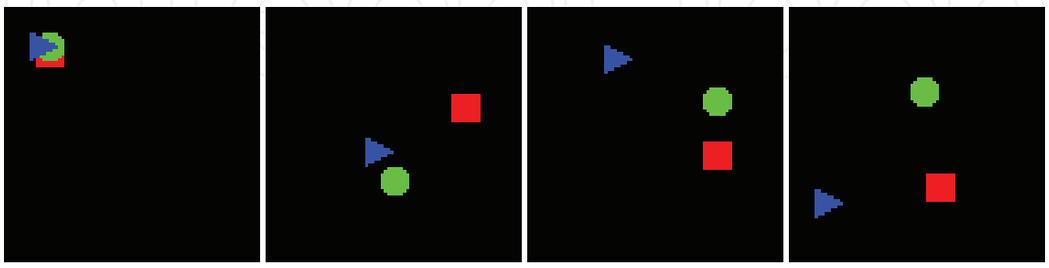


Fig. 6. Four example frames from the toy sequence. The full sequence is one hundred frames long, Each frame has a size of 80x80 pixels.

All three models are able to reconstruct their inputs from the latent variables with high confidence, but the way the input is encoded is very different. The learned receptive field

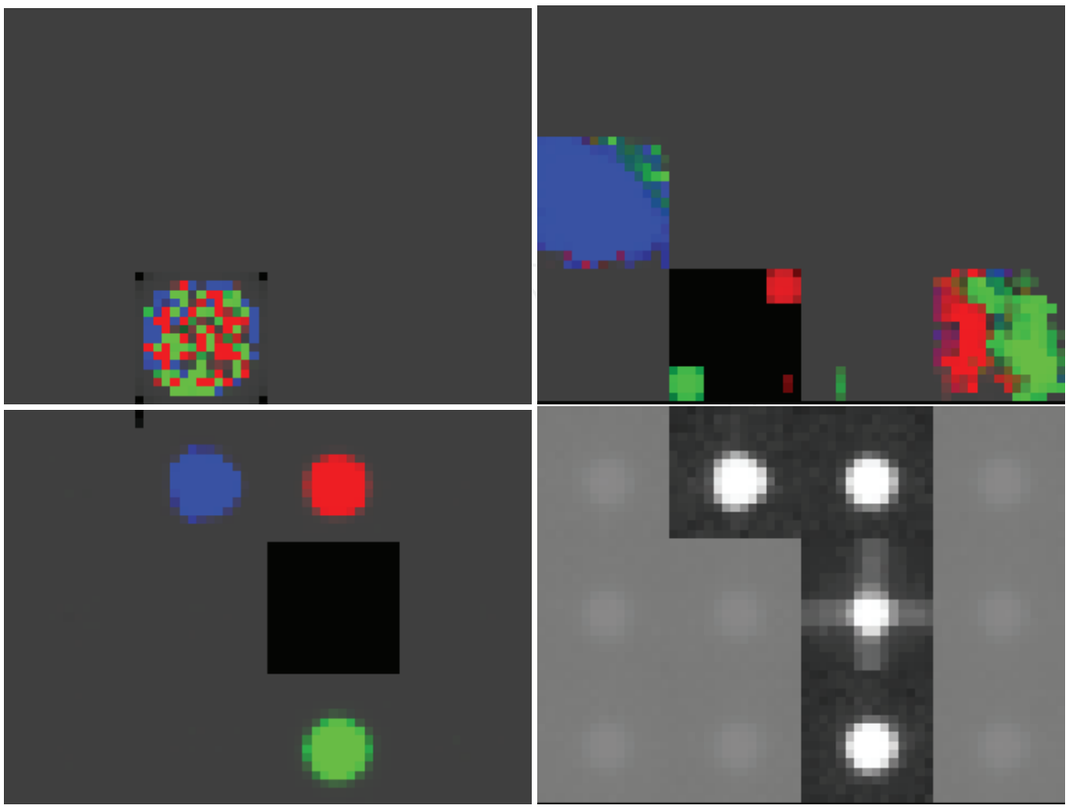


Fig. 7. Results of training the three model configurations. *upper left:* Receptive fields learned by the model with no embedded gate-MRF. No structure has been learned such that the model has to reconstruct the image pixel by pixel. *upper right:* Receptive fields learned by the model with embedded gate-MRF but no mask bits. There is more structure in the receptive fields than with no gate-MRF, but the model fails to separate the objects from each other.*lower left and right:* Receptive fields and mask bit distributions learned by the model with gate-MRF and mask bits. This model has learned a superpixel-decomposition of the image, i.e. the receptive fields and masks represent blobs of the three colors and black (the cross-shape superimposed on the mask for the black receptive field is due to border effects).

| model | $\langle H(gates \mid object) \rangle$ [bits] | normalized mutual information |
|---|---|---|
| 1 | 5.78 | 0 |
| 2 | 2.58 | 0.26 |
| 3 | 2.23 | 0.99 |

Table 2. Overview over the three model configurations analyzed. A cross means the corresponding feature is active in this configuration, a dash means it is not active.

together with the learned mask distribution, where applicable, are shown in figure 7. The simplest model with no gate MRF and no mask bits actually only learns a single receptive field containing pixels of all four colors. Thus all latent variables have the same setting and the gates assume settings effectively picking a pixel of the right color by selecting a latent variable with the receptive field aligned in the right way and each latent variable only models a few pixels, often only one, and each shape is modeled by many latent variables, thus no object concept is formed. When I introduce the gate MRF, enforcing smoothness in the gate field,


Fig. 8. Four examples of the training images used for the barcode localization task.

several receptive fields are learned and some smoothness in them emerges, but the model fails to fully separate the four objects from each others and the receptive fields still contain several colors. When the mask bits are introduced conditioned on the latent variables the model learns a super-pixel representation of the image: Four receptive fields with medium sized blobs of a each color or black emerge, and the blob-shape is encoded in the conditional mask bit distribution. While this is not quite what was hoped for the resulting code is a reasonable representation of the data, as can be seen by evaluating the average gating entropy per object, which I define by the entropy of the multinomial distribution over parents for the objects induced by the gates, and the normalized mutual information between the object identity and the latent variables, summarized in table 2 for all three experiments: Using the gate MRF and the variable receptive field shape each object can be encoded by around two bits with high fidelity.

### 6.2 Code tag localization

Here I apply the model to the task of locating code tags in images of items in a logistics context. The goal is to automatically identify the items by the tags placed on them from a camera image. Examples of such images are shown in figure 8. Labels for this task where generated automatically by the slow lookup procedure and hand-corrected to yield a dataset of 380 labeled images. The localization serves as a first step. In the second step the located tags are looked up in a separately maintained database. Since this step is quite slow, especially when the database is large, accurate localization is crucial.
For this task I preprocess the images by extracting two kinds of features:

- local Gabor wavelet decomposition in two scales and eight orientations
- on-center off-surround filters serving as local brightness indicators, implemented by heavily compressing the output of a local bandpass filter

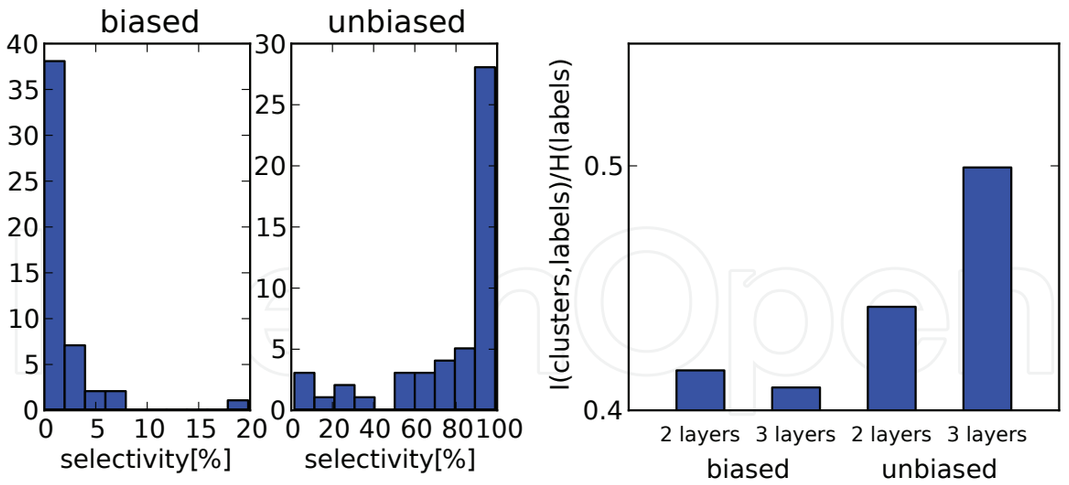



Fig. 9. *left and middle:*Histogram of cluster selectivities towards the code tags on the second layer for different bias conditions, illustrating the effect of training set bias. On the left side the trainig set bias was left uncorrected, such that background data was effectivey weighed ten times more than the targets. On the right this bias was removed by reweighing the M-step statistics so that both classes where weighed equally. The x-axis in each case shows the percentage of code tag patches relative to all input patterns assigned to a certain cluster. *right:*Mutual information between the output layer states and the labels for different settings, normalized w.r.t. to the label entropy. When the bias in the labels is removed via reweighing the additional layer helps the network in extracting the relevant information.

The resolution of these features is reduced by local maximum-pooling in each feature channel and non-overlapping two by two windows. These features are then concatenated for each location and fed into a three layer hierarchical network. This network learns to represent all of the images, not only the target regions, due to the generative nature of the model. Since during the learning stage of the hierarchy the labels are not used a strong bias in the dataset can cause a poor representation of the underrepresented class. In this example the code tags occupy only relatively small regions of the images and thus run into danger of being under-represented. To highlight the effect of training set bias I analyze the selectivity of cluster labels on the second layer for one or the other class. In figure 9 histograms of this selectivity are shown with a naive learning strategy, and one where the statistics used to update the parameters during the VBM-step where reweighed to remove the bias. As can be seen the naive strategy leaves only few, poorly selective clusters. In this case even discriminative post training can not be expected to yield very good results. On the other hand when the bias is removed the selectivity becomes more biased towards the target class (the code tags in this example). This can be understood by noting that most of the background is relatively simple, and can thus be represented by few clusters. The right-most plot in the same figure further highlights the benefit of de-biasing by showing the mutual information of cluster labels on the second and third layer with the target labels. As can be seen with the bias the second layer representation is so poor, that the third layer fails to build a better representation of the target class than the second layer. In the un-biased case this effect is gone and the third layer succeeds in improving the representation. For classification the hierarchy is trained layer by layer on eighty percent of the images. then the output of the third layer is fed into a naive Bayes classifier to learn a mapping to the target labels. The whole setup is illustrated in figure 10. The resulting system is then fine tuned using discriminative gradient descent. Figure shows the receiver operator

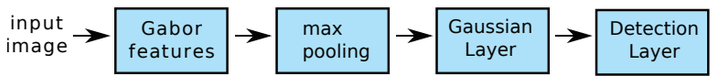



Fig. 10. Processing chain for the code tag localization task.

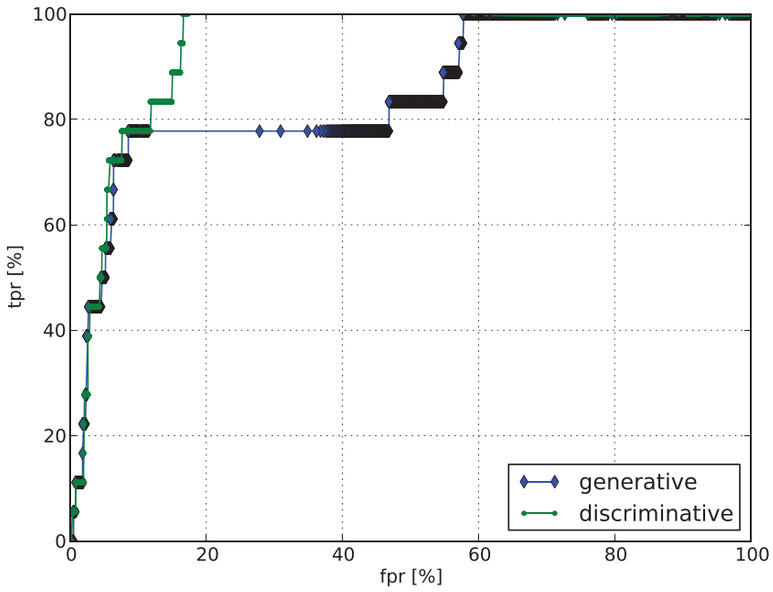



Fig. 11. ROC curves of the code tag detector in the biased and unbiased setting after generative and discriminative training.

Fig. 12. Positive examples from the INRIA pedestrians database used for training the pedestrian detector.

characteristic (ROC) curves for the remaining twenty percent of the images after generative and discriminative training.

### 6.3 Pedestrian detection

Pedestrian detection is an important application for object detection algorithms. As low level features I use the histograms of oriented gradients as established in Dalal & Triggs (2005), and also the training set introduced in this publication[4] ( see figures 12 and 13 for examples). I use a cell-size of six pixels, a block size of 3 cells, and a block-overlap of one cell. These features are used as the input features for a classifier based on the hierarchical image model (see figure 14): These features are first fed into a 3-layer network. The lowest layer of this network has

[4] http://pascal.inrialpes.fr/data/human/

Fig. 13. Negative examples from the INRIA pedestrians database used for training the pedestrian detector.

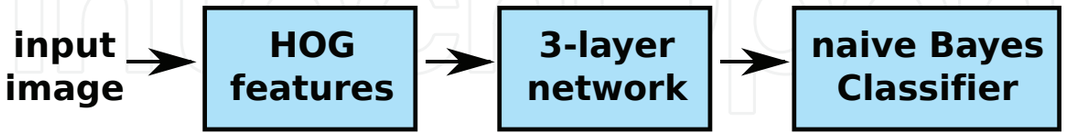



Fig. 14. Processing chain for pedestrian detection. The features are those suggested in Dalal & Triggs (2005).

receptive fields covering two by two blocks, i.e. four blocks total, with neighboring fields overlapping by one block. The next layer has receptive fields of four by four units. The top layer finally has seven by two receptive fields to cover a whole pedestrian (which in this dataset are standardized to 64x128 pixels). The receptive fields of the last layer overlap maximally to achieve relatively dense scanning for pedestrians in the input image.
Each layer of the network is trained generatively on the training set (see ) to convergence using VBEM before the next layer is trained. During this training the negative examples are weighed less than the positive examples by a factor of two hundred to avoid too detailed learning of the negative data. The resulting top layer cluster labels for each input window are then fed into a naive Bayes classifier to learn a mapping to the class labels (i.e. pedestrian/non-pedestrian). Subsequently the classifier and the hierarchical network are refined discriminatively using gradient descent.
In figure I show the resulting precision-recall-curve of this setup on the test set after generative training and after the subsequent discriminative refinement. The results are comparable to those of kernelized R-HOG as presented in Dalal & Triggs (2005), though somewhat weaker in the high precision regime. The discriminative training increases the recall but further looses precision. It would be worth investigating how this precision loss may be prevented in a deep hierarchy such as this for applications that need it.

### 6.4 Locating faces

Here I demonstrate how the hierarchical model can be used to locate faces in images. As training I use a relatively small dataset in which the faces appear isolated against a blank wall. Furthermore during training I do not tell the model the location of the faces in the image. As training data I use the frontal views of the faces in the CBCL faces training dataset[5] shown in figure 16. Obviously this is a very small dataset, so correctly modeling parameter uncertainty is important to avoid overfitting. I train a two layer model. The lowest layer consists of a simple categorical layer with a Gaussian emission model. This layer operates on Gabor features with a bandwidth of one octave at eight orientations and two scales, where the finest scale has a wavelength of five pixels which are max-pooled by a factor of four [6].

[5] http://cbcl.mit.edu/software-datasets/heisele/facerecognition-database.html
[6] By "max-pooling" I mean downsampling where the max instead of the mean operation is used to summarize the input pixels flowing into one output pixel.

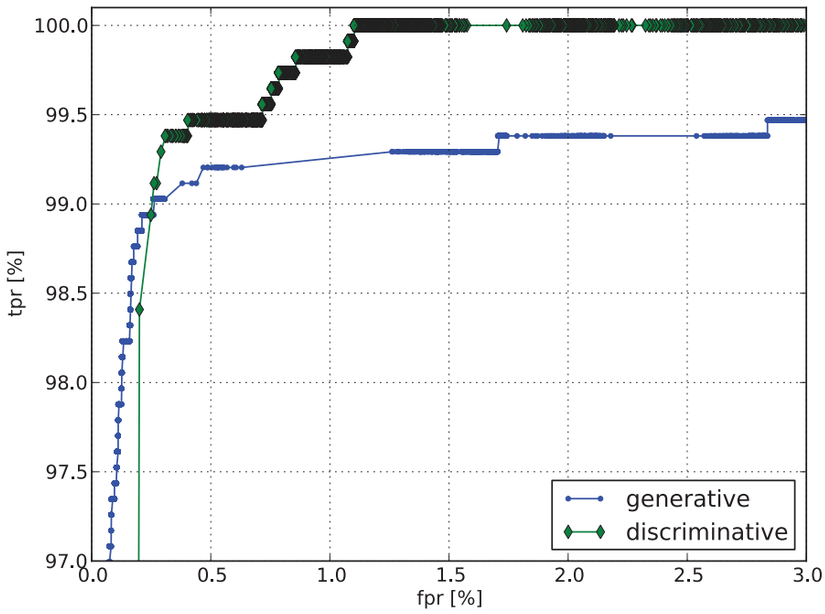



Fig. 15. ROC characteristics of the pedestrian detector after generative training and after discriminiative optimization .

Fig. 16. Positive examples from the CBCL faces database used for training the face model.

The layer has a receptive field size of four by four with halve-overlapping receptive fields. I then use shift and scale-invariant VBEM training as described in section 5.5 to learn a single multivariate multinomial representation of a frontal face, using the output of the first layer as input. To improve this model I use conjugate gradients to do feature selection as described in section 5.6, using ten percent of the clutter images from the Caltech 256 dataset [7] as negative examples. To test the model I use it to locate the faces in the first one hundred images from the Caltech faces database [8] which shows frontal views of faces in complex environments and different lighting conditions. After feature selection the model correctly locates all but seven of the faces (see figure 19), six of which are from the same person, who seems to be badly

[7] http://www.vision.caltech.edu/Image_Datasets/Caltech256/
[8] `http://www.vision.caltech.edu/Image_datasets/faces/faces.tar`

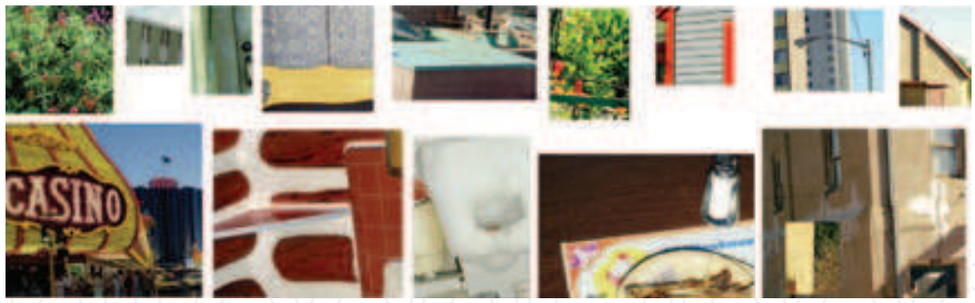



Fig. 17. Some of the negative examples from the Caltech256 database used for training feature selection.

Fig. 18. The seven failures when locating faces in the first one hundred images from the Caltech fasesdataset.Obviouslythe network mostly has problems locating one specific person.

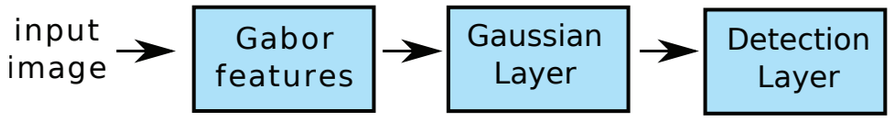



Fig. 19. The seven failures when locating faces in the first one hundred images from the Caltech fasesdataset.Obviouslythe network mostly has problems locating one specific person.

represented by the training set. Without feature selection the localization rate drops to about eighty percent.

## 7. Conclusion

In conclusion I introduced the Bayesian hierarchical image model for learning of statistics of arbitrary images. I have explained the motivation and basic ideas behind the generative modeling approach. I have reviewed Bayesian inference techniques and how they are applied to this model. I have explained how Bayesian techniques can be used to learn such a model with robustness to overfitting, while discrimiantive gradient descent can be used to fine tune the classification performance. I have also introduced an extension to the basic model that leading to a better representation of images by introducing automatic segmentation into the model.

In the experiments I have shown the performance of the model in various tasks. To present the behavior of the segmentation I have turned to a toy example where the details of the model behavior could be readily analyzed. The basic model in turn has been applied to several real world examples. There I showed the importance of managing training set bias by reweighing the learning statistics. The ability to do this is a consequence of the truly Bayesian treatment of

the learning process. Also the results on the competitive INRIA pedestrian detection dataset shows that this approach is a valid competitor for more classical approaches such as support vector machines.
Interesting directions from here would be the analysis and possible improvement of transformation invariance, and the application of the segmentation extension to real world problems.

## 8. References


Attias, H. (1999). Inferring parameters and structure of latent variable models by variational bayes, *Proceedings of Fifteenth Conference on Uncertainty in Artificial Intelligence*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, Springer-Verlag.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

Chalupa, L. M. & Werner, J. S. (eds) (2003). *The Visual Neurosciences*, MIT Press.

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection, *in* C. Schmid, S. Soatto & C. Tomasi (eds), *International Conference on Computer Vision & Pattern Recognition*, Vol. 2, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, pp. 886–893.
URL: *http://lear.inrialpes.fr/pubs/2005/DT05*

Dean, T. (2006). Scalable inference in hierarchical generative models, *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*.

Dempster, A., Laird, N. & Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm, *Journal of the Royal Statistical Society, Series B* 39(1): 1–38.

Eddy, S. R. (2004). What is dynamic programming?, *Nature Biotechnology* (22): 909–910.

Fink, D. (1995). A compendium of conjugate priors. in progress report: Extension and enhancement of methods for setting data quality objectives., *Technical report*, Cornell University.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biol. Cyb.* V36(4): 193–202.
URL: *http://dx.doi.org/10.1007/BF00344251*

Hadsell, R., Chopra, S. & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping, *Proc. CVPR*, IEEE Press.

Hinton, G. E. (2007). Learning multiple layers of representation, *TRENDS in Cognitive Sciences* 11: 428.

J. M. Winn, C. M. B. (2005). Variational message passing, *J. Mach. L. Res.* 6: 558–590.

Lasserre, J., Lasserre, J., Bishop, C. & Minka, T. (2006). Principled hybrids of generative and discriminative models, *in* C. Bishop (ed.), *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 87–94.

LeCun, Y., Bottou, L., Orr, G. B. & Mueller, K.-R. (1998). Efficient backprop, *Lecture Notes in Computer Science* 1524: 9–??
URL: *citeseer.ist.psu.edu/lecun98efficient.html*

LeCun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D., Howard, R. E. & Hubbard, W. (1989). Handwritten digit recognition: Applications of neural net chips and automatic learning, *IEEE Comm.* pp. 41–46. invited paper.

Minka, T. & Winn, J. (2008). Gates: A graphical notation for mixture models, *Technical Report MSR-TR-2008-185*, Microsoft Research.

Mutch, J. & Lowe, D. G. (2006). Multiclass object recognition with sparse, localized features, *Proc. CVPR* pp. 11–18.

Onsager, L. (1944). Crystal statistics. i. a two-dimensional model with an order-disorder transition, *Phys. Rev.* 65(3-4): 117–149.

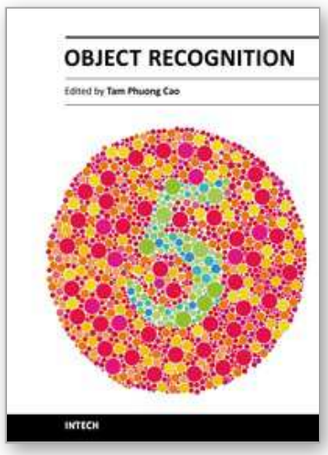
Pearl, J. (1982). Reverend bayes on inference engines: A distributed hierarchical approach.

Rolls, E. T. & Deco, G. (2002). *The Computational Neuroscience of Vision*, Oxford University Press.

Serre, T., Kouh, M., Cadieu, C., Knoblich, U. & G. Kreiman, T. P. (2005). A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex, *CBCL Memo 259*.

Wellman, M. P. & Henrion, M. (1993). Explaining "explaining away", *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 15: 287–292.

Wersing, H. & Korner, E. (2003). Learning Optimized Features for Hierarchical Models of Invariant Object Recognition, *Neural Comp.* 15(7): 1559–1588.
URL: *http://neco.mitpress.org/cgi/content/abstract/15/7/1559*




**Object Recognition**

Edited by Dr. Tam Phuong Cao

ISBN 978-953-307-222-7
Hard cover, 350 pages
**Publisher** InTech
**Published online** 01, April, 2011
**Published in print edition** April, 2011

Vision-based object recognition tasks are very familiar in our everyday activities, such as driving our car in the correct lane. We do these tasks effortlessly in real-time. In the last decades, with the advancement of computer technology, researchers and application developers are trying to mimic the humanâ€™s capability of visually recognising. Such capability will allow machine to free human from boring or dangerous jobs.

**How to reference**

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Daniel Oberhoff (2011). Hierarchical Bayesian Image Models, Object Recognition, Dr. Tam Phuong Cao (Ed.), ISBN: 978-953-307-222-7, InTech, Available from: http://www.intechopen.com/books/object-recognition/hierarchical-bayesian-image-models

INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen