Journal of Machine Learning Research 12 (2011) 3071-3096 Submitted 10/10; Revised 7/11; Published 11/11

In All Likelihood, Deep Belief Is Not Enough

Lucas Theis LUCAS.THEIS@TUEBINGEN.MPG.DE
Sebastian Gerwinn SEBASTIAN.GERWINN@TUEBINGEN.MPG.DE
Fabian Sinz FABIAN.SINZ@TUEBINGEN.MPG.DE
Matthias Bethge MATTHIAS.BETHGE@TUEBINGEN.MPG.DE
Werner Reichardt Centre for Integrative Neuroscience
Bernstein Center for Computational Neuroscience
Max Planck Institute for Biological Cybernetics
Spemannstraße 41, 72076 Tübingen, Germany

Editor: Yoshua Bengio

Abstract

Statistical models of natural images provide an important tool for researchers in the fields of machine learning and computational neuroscience. The canonical measure to quantitatively assess and compare the performance of statistical models is given by the likelihood. One class of statistical models which has recently gained increasing popularity and has been applied to a variety of complex data is formed by deep belief networks. Analyses of these models, however, have often been limited to qualitative analyses based on samples due to the computationally intractable nature of their likelihood. Motivated by these circumstances, the present article introduces a consistent estimator for the likelihood of deep belief networks which is computationally tractable and simple to apply in practice. Using this estimator, we quantitatively investigate a deep belief network for natural image patches and compare its performance to the performance of other models for natural image patches. We find that the deep belief network is outperformed with respect to the likelihood even by very simple mixture models.

Keywords: deep belief network, restricted Boltzmann machine, likelihood estimation, natural image statistics, potential log-likelihood

1. Introduction

When dealing with natural images, the choice of image representation is often crucial for achieving a good performance. Good generative models or classifiers, for example, can be easier to realize in terms of more complex features such as edges than in terms of raw pixel intensities. Several facts point to the advantage of using hierarchical representations for encoding natural images over non-hierarchical ones. Multiple layers of representations allow for the use of simpler transformations, each solving only a subproblem, as well as for a more efficient implementation by enabling the reuse of low-level features in the realization of higher-level features. Further motivation for hierarchical representations comes from the hierarchical organization of the brain (Felleman and van Essen, 1991) and the hierarchical organization of the concepts surrounding us. The idea of using hierarchical image representations in supervised as well as unsupervised tasks is now several decades old (e.g., Selfridge, 1958; Fukushima, 1980; LeCun et al., 1989). However, only the emergence of recent training methods has made them competitive across many supervised learning tasks and led to a renewed surge of interest in hierarchical image representations. Despite this success in supervised tasks, no probabilistic model has been conclusively shown to exhibit state-of-the-art performance as a *generative* model and at the same time benefit strongly from hierarchical image representations.

The most prominent example of the recent research in hierarchical image modeling is the research into a class of hierarchical generative models called *deep belief networks*. Deep belief networks were introduced by Hinton and Salakhutdinov (2006); Hinton et al. (2006) together with a greedy learning rule as an approach to the long-standing challenge of training *deep* neural networks, that is, hierarchical neural networks such as multi-layer perceptrons. The existence of an efficient learning rule has made them become attractive not only for pretraining multi-layer perceptrons, but also for density estimation and other inherently unsupervised learning tasks. In supervised tasks, they have been shown to learn representations which outperform many competing representations when employed, for example, in character recognition (Hinton et al., 2006) or speech recognition (Mohamed et al., 2009). In unsupervised tasks, they have been applied to a wide variety of complex data sets such as patches of natural images (Osindero and Hinton, 2008; Ranzato et al., 2010a; Ranzato and Hinton, 2010; Lee and Ng, 2007), motion capture recordings (Taylor et al., 2007) and images of faces (Susskind et al., 2008). When applied to natural images, deep belief networks have been shown to develop biologically plausible features (Lee and Ng, 2007) and samples from the model were shown to adhere to certain statistical regularities also found in natural images (Osindero and Hinton, 2008). Examples of natural image patches and features learned by a deep belief network are presented in Figure 1.

An important measure to assess the generative performance of a probabilistic model is the likelihood. The likelihood allows us to objectively compare the density estimation performance of different models. Given two model instances with equal a priori probability, the ratio of their likelihoods with respect to a set of data samples tells us everything we need to know to decide which of the two models is more likely to have generated the data set. Further motivation for the likelihood stems from coding theory. For densities p and q, the negative expected log-likelihood represents the *cross-entropy* term of the Kullback-Leibler (KL) divergence,

$$D_{\mathrm{KL}}[p(x)||q(x)] = -\sum_x p(x)\log q(x) - \mathbf{H}[p(x)],$$

which is always non-negative and zero if and only if p and q are identical. The cross-entropy represents the coding cost of encoding samples drawn from p with a code that would be optimal for samples drawn from q. Correspondingly, the KL-divergence represents the additional coding cost created by using an optimal code which assumes the distribution of the samples to be q instead of p.

The expected negative log-likelihood, or cross-entropy, quantifies the amount of correlations captured by a statistical model. A model with a minimal cross-entropy would, at least in principle, be able to predict missing information from partially observed input in an optimal manner. In this sense, the likelihood can be understood as a measure of scene understanding if a model is applied to natural scenes.

Finally, the likelihood allows us to directly examine the success of training when maximum likelihood learning is employed. Even when the ultimate goal is classification, deep belief networks and related unsupervised feature learning approaches are optimized with respect to the likelihood. Evaluating the likelihood is therefore also important to assess the success of pretraining and for fine-tuning hyperparameters. Unfortunately, the likelihood of deep belief networks is in general computationally intractable to evaluate.

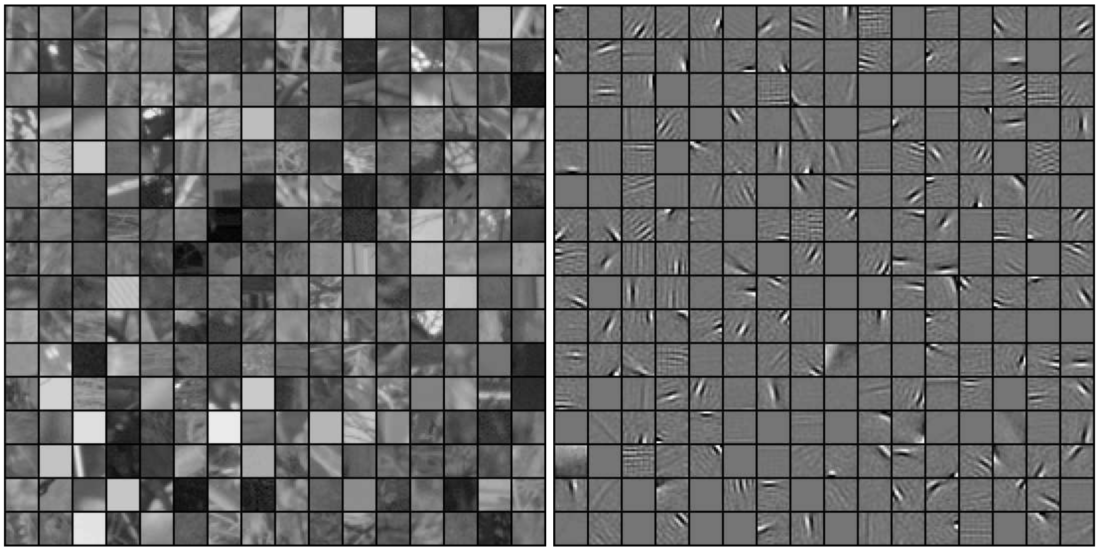

Figure 1: *Left:* Natural image patches sampled from the van Hateren dataset (van Hateren and van der Schaaf, 1998). *Right:* Filters learned by a deep belief network trained on whitened image patches.

In this article, we set out to test the performance of a deep belief network by evaluating its likelihood. After reviewing the relevant aspects of deep belief networks, we will derive a new consistent estimator for their likelihood and demonstrate the estimator's applicability in practice. We will investigate a particular deep belief network's capability to model the statistical regularities found in natural image patches. We will show that the deep belief network under study is not particularly good at capturing the statistics of natural image patches as it is outperformed with respect to the likelihood even by very simple mixture models. We will furthermore show that adding layers to the network has only a small effect on the overall performance of the model if the first layer is trained well enough and offer possible explanations for this observation by analyzing a best-case scenario of the greedy learning procedure commonly used for training deep belief networks.

2. Models

In this section we will review the statistical models used in the remainder of this article and discuss some of their properties relevant for estimating the likelihood of deep belief networks (DBNs). Throughout this section, the goal of applying statistical models is assumed to be the approximation of a particular distribution of interest, the *data distribution*. We will denote this distribution by $\tilde{p}$.

2.1 Boltzmann Machines

A *Boltzmann machine* is a potentially fully connected *undirected graphical model* with binary random variables. Its probability mass function is a Boltzmann distribution over 2^k binary states

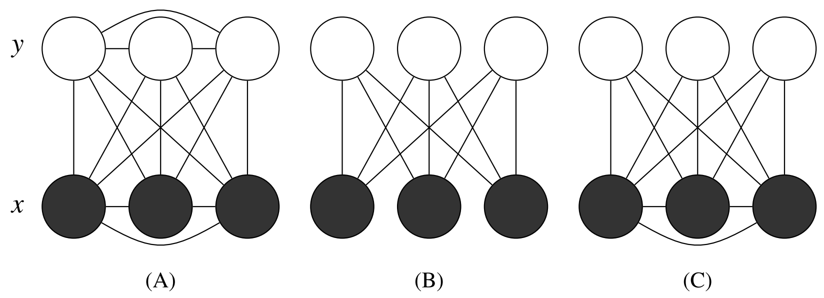

Figure 2: Boltzmann machines with different constraints on their connectivity. Filled nodes denote visible variables, unfilled nodes denote hidden variables. *A:* A fully connected Boltzmann machine. *B:* A restricted Boltzmann machine. *C:* A semi-restricted Boltzmann machine, which in contrast to RBMs also allows connections between the visible units.

$s \in \{0,1\}^k$ which is defined in terms of an *energy function* E,

$$q(s) = \frac{1}{Z}\exp(-E(s)), \quad Z = \sum_s \exp(-E(s)),$$

where E is given by

$$E(s) = -\frac{1}{2}s^\top Ws - b^\top s = -\frac{1}{2}\sum_{i,j} s_i w_{ij} s_j - \sum_i s_i b_i$$

and depends on a symmetric weight matrix $W \in \mathbb{R}^{k\times k}$ with zeros on the diagonal, $w_{ii} = 0$ for all $i = 1,\dots,k$, and bias terms $b \in \mathbb{R}^k$. Z is called *partition function* and ensures the normalization of q. In the following, unnormalized distributions will be marked with an asterisk:

$$q^*(s) = Zq(s) = \exp(-E(s)).$$

Of particular interest for building DBNs are *latent variable Boltzmann machines*, that is, Boltzmann machines for which the states s are only partially observed (Figure 2). We will refer to states of observed or visible random variables as x and to states of unobserved or hidden random variables as y, such that $s = (x,y)$.

Maximum likelihood (ML) learning can be implemented by following the gradient of the log-likelihood. In Boltzmann machines, this gradient is conceptually simple yet computationally hard to evaluate. The gradient of the expected log-likelihood with respect to some parameter θ of the energy function is (e.g., Salakhutdinov, 2009):

$$\mathbf{E}_{\tilde{p}(x)}\left[\frac{\partial}{\partial\theta}\log q(x)\right] = \mathbf{E}_{q(x,y)}\left[\frac{\partial}{\partial\theta}E(x,y)\right] - \mathbf{E}_{\tilde{p}(x)q(y|x)}\left[\frac{\partial}{\partial\theta}E(x,y)\right]. \quad (1)$$

The first term on the right-hand side of this equation is the expected gradient of the energy function when both hidden and visible states are sampled from the model, while the second term is the expected gradient of the energy function when the hidden states are drawn from the conditional distribution of the model, given a visible state drawn from the data distribution, $\tilde{p}(x)$.

Evaluating these expectations, however, is computationally intractable for all but the simplest models. Even approximating the expectations with Monte Carlo methods is typically very slow (Long and Servedio, 2010). Two measures can be taken to make learning in Boltzmann machines feasible: constraining the Boltzmann machine in some way, or replacing the likelihood with a simpler objective function. The latter approach led to the introduction of the contrastive divergence (CD) learning rule (Hinton, 2002) which represents a tractable approximation to ML learning: In CD learning, the expectation over the model distribution $q(x,y)$ is replaced by an expectation over

$$q_{\text{CD}}(x,y) = \sum_{x_0,y_1} \tilde{p}(x_0) q(y_1 \mid x_0) q(x \mid y_1) q(y \mid x),$$

from which samples are obtained by taking a sample x_0 from the data distribution, updating the hidden units, updating the visible units, and finally updating the hidden units again, while in each step keeping the respective set of other variables fixed. This corresponds to a single sweep of Gibbs sampling through all random variables of the model plus an additional update of the hidden units. If instead n sweeps of Gibbs sampling are used, the learning procedure is generally referred to as CD(n) learning. In the limit of large n, ML learning is regained (Salakhutdinov, 2009). An improved sampling scheme is offered by *persistent contrastive divergence* (PCD), in which the Markov chain is initialized not with a sample from the data distribution, but with the state of the Markov chain at the previous update of the gradient (Younes, 1989; Tieleman, 2008).

2.2 Restricted Boltzmann Machines

The first expectation on the right-hand side of Equation 1 can be made analytically tractable by constraining the energy function such that no direct interaction between two visible units or two hidden units is possible (Smolensky, 1986; Hinton, 2002),

$$E(x,y) = -x^\top Wy - b^\top x - c^\top y.$$

Such a model is called a restricted Boltzmann machine (RBM). The corresponding graph has no connections between the visible units and no connections between the hidden units (Figure 2). Importantly, the unnormalized marginal distributions $q^*(x)$ and $q^*(y)$ can now be computed analytically by integrating out the respective other set of variables. The unnormalized marginal distribution of the visible units becomes

$$q^*(x) = \exp(b^\top x) \prod_j (1 + \exp(w_j^\top x + c_j)). \tag{2}$$

Two related models also used in this article are the *Gaussian RBM* (GRBM) (Salakhutdinov, 2009) and the *semi-restricted Boltzmann machine* (SRBM) (Osindero and Hinton, 2008). The GRBM employs continuous visible units and binary hidden units and can thus be used to model continuous data. Its energy function is given by

$$E(x,y) = \frac{1}{2\sigma^2}||x-b||^2 - \frac{1}{\sigma} x^\top Wy - c^\top y.$$

A somewhat more general definition allows a different σ for each individual visible unit (Salakhutdinov, 2009). The conditional distribution $q(x \mid y)$ of a GRBM is a multivariate Gaussian distribution,

$$q(x \mid y) = \mathcal{N}(x; \sigma Wy + b, \sigma^2 I).$$

Each binary state of the hidden units encodes one mean, while σ controls the variance of each Gaussian and is the same for all hidden units. The GRBM can therefore be interpreted as a mixture of an exponential number of Gaussian distributions with fixed, isotropic covariance and parameter sharing constraints.

In an SRBM, only the hidden units are constrained to have no direct connections to each other while the visible units are unconstrained (Figure 2). Analytic expressions are therefore only available for $q^*(x)$ but not for $q^*(y)$ and the visible units are no longer independent given a state for the hidden units.

2.3 Deep Belief Networks

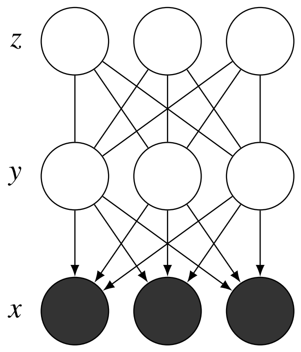

Figure 3: A graphical model representation of a two-layer deep belief network composed of two RBMs. The connections of the first layer are directed.

DBNs (Hinton and Salakhutdinov, 2006) are hierarchical generative models composed of several layers of RBMs or one of their generalizations. Let $q(x,y)$ and $r(y,z)$ be the densities of two RBMs over visible states x and hidden states y and z. Then the joint probability mass function of a two-layer DBN is defined to be

$$p(x,y,z) = q(x \mid y)r(y,z).$$

The resulting model is best described not as a deep Boltzmann machine but as a graphical model with undirected connections between y and z and directed connections between x and y (Figure 3). This definition can be recursively extended to DBNs with three or more layers by replacing $r(y,z)$ with another DBN. DBNs with an arbitrary number of layers have been shown to be universal approximators even if the number of hidden units in each layer is fixed to the number of visible units (Sutskever and Hinton, 2008). DBNs are easily generalized by allowing more general models as layers, such as the GRBM and the SRBM.

The learning procedure introduced by Hinton et al. (2006) for training DBNs makes two approximations to ML learning. The first approximation is made by training the DBN in a greedy manner: After the first layer of the model has been trained to approximate the data distribution, its parameters are fixed and only the parameters of the second layer are optimized. If θ is a parameter of the second-layer density r, the gradient of the DBN's log-likelihood with respect to θ is

$$\frac{\partial}{\partial\theta}\log p(x) = \sum_y p(y \mid x)\frac{\partial}{\partial\theta}\log r(y). \tag{3}$$

However, exact sampling from the posterior distribution $p(y \mid x)$ is difficult. In order to make the training feasible, the posterior distribution is replaced by the factorial distribution $q(y \mid x)$. Training the DBN in this manner optimizes a variational lower bound on the log-likelihood (Hinton et al., 2006),

$$\sum_y q(y \mid x)\log r(y) \leq \log p(x) + const, \tag{4}$$

where *const* is constant in θ, which is a parameter of r. Taking the derivative of the left-hand side of Equation 4 with respect to θ yields (3) with the posterior distribution $p(y \mid x)$ replaced by $q(y \mid x)$. The greedy learning procedure can be generalized to more layers by training each additional layer to approximate the distribution obtained by conditionally sampling from each layer in turn, starting with the lowest layer.

After finishing the greedy training, Hinton et al. (2006) suggested to use the *wake-sleep algorithm* (Hinton et al., 1995) to fine-tune the parameters. Like the greedy algorithm, the wake-sleep algorithm optimizes a lower bound on the log-likelihood of the model, but in contrast optimizes all parameters concurrently. In addition, a separate set of RBMs is used and adapted to more closely approximate the DBN's posterior distribution over hidden units.

3. Likelihood Estimation

In this section, we will discuss the problem of estimating the likelihood of a two-layer DBN with joint density

$$p(x,y,z) = q(x \mid y)r(y,z). \tag{5}$$

That is, for a given visible state x, to estimate the value of

$$p(x) = \sum_{y,z} q(x \mid y)r(y,z).$$

We will later generalize this problem to more layers. As before, $q(x,y)$ and $r(y,z)$ refer to the densities of two RBMs.

Two difficulties arise when dealing with this problem in the context of DBNs. First, $r(y,z)$ depends on a partition function Z_r whose exact evaluation requires integration over an exponential number of states. Second, despite being able to integrate analytically over z, even computing just the unnormalized likelihood still requires integration over an exponential number of hidden states y,

$$p^*(x) = \sum_y q(x \mid y)r^*(y).$$

After briefly reviewing previous approaches to resolving these difficulties, we will propose an unbiased estimator for $p^*(x)$, its contribution being a possible solution to the second problem, and discuss how to construct a consistent estimator for $p(x)$ based on this result.

3.1 Importance Sampling

Since our estimator relies on importance sampling, we briefly review it here. Importance sampling is a Monte Carlo integration method for unbiased estimation of expectations (MacKay, 2003) and is based on the following observation: Let s be a density with $s(x) > 0$ whenever $q^*(x) > 0$ and let $w(x) = \frac{q^*(x)}{s(x)}$, then

$$\sum_x q^*(x) f(x) = \sum_x s(x) \frac{q^*(x)}{s(x)} f(x) = \mathbf{E}_{s(x)}[w(x) f(x)]$$

for any function f. s is called a *proposal distribution* for q and $w(x)$ is called *importance weight*. If $f(x) = 1$ for all x, we get

$$\mathbf{E}_{s(x)}[w(x)] = \sum_x s(x) \frac{q^*(x)}{s(x)} = Z_q. \tag{6}$$

Estimates of the partition function Z_q can therefore be obtained by drawing samples $x^{(n)}$ from a proposal distribution and averaging the resulting importance weights $w(x^{(n)})$. It was pointed out in Minka (2005) that minimizing the variance of the importance sampling estimate of the partition function (6) is equivalent to minimizing an α-divergence[1] between the proposal distribution s and the true distribution q. Therefore, for an estimator to work well in practice, s should be both close to q and easy to sample from.

3.2 Previous Work

The following is a brief summary of existing approaches to approximating the likelihood of RBMs and DBNs.

3.2.1 ANNEALED IMPORTANCE SAMPLING

Salakhutdinov and Murray (2008) have shown how *annealed importance sampling* (AIS) (Neal, 2001) can be used to estimate the partition function of an RBM. AIS tries to circumvent some of the problems associated with finding a suitable proposal distribution. For a sequence of proposal distributions $s_1, \ldots, s_n$ and corresponding transition operators $T_1, \ldots, T_n$, one can show that

$$Z_q = \sum_x s_n(x_{n-1}) T_{n-1}(x_{n-2}; x_{n-1}) \cdots T_1(x_0; x_1) \frac{s^*_{n-1}(x_{n-1})}{s_n(x_{n-1})} \cdots \frac{q^*(x_0)}{s^*_1(x_0)},$$

where the sum integrates over all $x = (x_0, \ldots, x_{n-1})$. Hence, in order to estimate the partition function, we can draw independent samples x_{n-1} from a simple distribution s_n, use the transition operators to generate samples $x_{n-2}, \ldots, x_0$ from increasingly complex distributions, and average the resulting product of fractions given in the preceding equation. For details on how to choose the intermediate distributions and transition operators, see Salakhutdinov and Murray (2008).

1. With $\alpha = 2$. α-divergences are a generalization of the KL-divergence.

3.2.2 ESTIMATING LOWER BOUNDS

In Salakhutdinov and Murray (2008) it was also shown how estimates of a lower bound on the log-likelihood,

$$\log p(x) \geq \sum_y q(y \mid x) \log \frac{r^*(y) q(x \mid y)}{q(y \mid x)} - \log Z_r \tag{7}$$

$$= \sum_y q(y \mid x) \log r^*(y) q(x \mid y) + \mathbf{H}[q(y \mid x)] - \log Z_r, \tag{8}$$

can be obtained, provided the partition function Z_r is given. This is the same lower bound as the one optimized during greedy learning (4). Since $q(y \mid x)$ is factorial, the entropy $\mathbf{H}[q(y \mid x)]$ can be computed analytically. The only term which still needs to be estimated is the first term on the right-hand side of Equation 8. This was achieved in Salakhutdinov and Murray (2008) by averaging over samples drawn from $q(y \mid x)$.

3.2.3 CONSISTENT ESTIMATES

In Murray and Salakhutdinov (2009), a rather elaborate sampling scheme was devised to give unbiased estimates for the inverse posterior probability $\frac{1}{p(y|x)}$ of some fixed hidden state y. These estimates were then used to get unbiased estimates of $p^*(x)$ by taking advantage of the fact $p^*(x) = \frac{p^*(x,y)}{p(y|x)}$. The corresponding partition function was estimated using AIS, giving rise to a consistent estimator. While the estimator's Markov chain was constructed such that arbitrarily short runs of the Markov chain result in unbiased estimates of $p^*(x)$, even a single step of the Markov chain is slow compared to sampling from $q(y \mid x)$, as it was done for the estimation of the lower bound (7).

3.3 A New Estimator for DBNs

The estimator we will introduce in this section shares the same formal properties as the estimator proposed in Murray and Salakhutdinov (2009), but will use samples drawn from $q(y \mid x)$. This will make it conceptually as simple and as easy to apply in practice as the estimator for the lower bound (7), while providing us with consistent estimates of $p(x)$.

Let $p(x,y,z)$ be the joint density of a DBN as defined in Equation 5. By applying Bayes' theorem, we obtain

$$p(x) = \sum_y q(x \mid y) r(y)$$

$$= \sum_y q(y \mid x) \frac{q(x)}{q(y)} r(y)$$

$$= \sum_y q(y \mid x) \frac{q^*(x)}{q^*(y)} \frac{r^*(y)}{Z_r}. \tag{9}$$

A natural choice for an estimator of $p(x)$ is therefore

$$\hat{p}_N(x) = \frac{1}{N} \sum_n \frac{q^*(x)}{q^*(y^{(n)})} \frac{r^*(y^{(n)})}{Z_r} \tag{10}$$

$$= q^*(x) \frac{1}{Z_r N} \sum_n \frac{r^*(y^{(n)})}{q^*(y^{(n)})}$$

where $y^{(n)} \sim q(y^{(n)} \mid x)$ for $n = 1, ..., N$. For RBMs, the unnormalized marginals $q^*(x), q^*(y)$ and $r^*(y)$ can be computed analytically (2). Also note that the partition function Z_r only has to be calculated once for all visible states we wish to evaluate. We are not aware of any work which tried to use this estimator to estimate the likelihood of DBNs. In the following, we will discuss and investigate its properties.

Under the assumption that the partition function Z_r is known, $\hat{p}_N(x)$ provides an unbiased estimate of $p(x)$ since the sample average is always an unbiased estimate of the expectation. However, Z_r is generally intractable to compute exactly so that approximations become necessary. If in the estimate (10) the partition function Z_r is replaced by an unbiased estimate $\hat{Z}_r$, then the overall estimate will tend to overestimate the true likelihood,

$$\begin{aligned}\mathbf{E}\left[\frac{\hat{p}_N^*(x)}{\hat{Z}_r}\right] &= \mathbf{E}\left[\frac{1}{\hat{Z}_r}\right]\mathbf{E}\left[\hat{p}_N^*(x)\right] \\ &\geq \frac{1}{\mathbf{E}\left[\hat{Z}_r\right]} p^*(x) = p(x),\end{aligned}$$

where $\hat{p}_N^*(x) = Z_r \hat{p}_N(x)$ is an unbiased estimate of the unnormalized density. The second step is a consequence of Jensen's inequality and the averages are taken with respect to $\hat{p}_N(x)$ and $\hat{Z}_r$, which are independent; x is held fix.

While the estimator loses its unbiasedness for unbiased estimates of the partition function, it still retains its consistency. Since $\hat{p}_N^*(x)$ is unbiased for all $N \in \mathbb{N}$, it is also asymptotically unbiased,

$$\operatorname*{plim}_{N\to\infty} \hat{p}_N^*(x) = p^*(x).$$

Furthermore, if $\hat{Z}_{r,N}$ for $N \in \mathbb{N}$ is a consistent sequence of estimators for the partition function, it follows that

$$\operatorname*{plim}_{N\to\infty} \frac{\hat{p}_N^*(x)}{\hat{Z}_{r,N}} = \frac{\operatorname*{plim}_{N\to\infty} \hat{p}_N^*(x)}{\operatorname*{plim}_{N\to\infty} \hat{Z}_{r,N}} = \frac{p^*(x)}{Z_r} = p(x).$$

Unbiased and consistent estimates of Z_r can be obtained using AIS (Salakhutdinov and Murray, 2008). Note that although the estimator tends to overestimate the true likelihood in expectation and is unbiased in the limit, it is still possible for it to underestimate the true likelihood most of the time. This behavior can occur if the distribution of estimates is heavily skewed.

An important question which remains is whether the estimator is also good in terms of statistical efficiency. The more efficient an estimator, the less samples are required to obtain reliable estimates of the likelihood. If we cast Equation 9 into the form of Equation 6, we see that our estimator performs importance sampling with proposal distribution $q(y \mid x)$ and target distribution $p(y \mid x)$, where $p(x)$ can be seen as the partition function of an unnormalized distribution $p(x,y)$. As mentioned earlier, the efficiency of importance sampling estimates of partition functions depends on how well the proposal distribution approximates the true distribution. Therefore, for the proposed estimator to work well in practice, $q(y \mid x)$ should be close to $p(y \mid x)$. Note that a similar assumption is made when optimizing the variational lower bound during greedy learning. The lower bound (4) can be shown to be equal to

$$\sum_y q(y \mid x) \log r(y) = \log p(x) - D_{KL}(q(y \mid x) || p(y \mid x)) + const, \tag{11}$$

```
Estimate $p(x_0)$
1: $\hat{p} \leftarrow q_1^*(x_0)$
2: for $l = 1$ to $L - 1$ do
3:     $x_l \sim q_l(x_l \mid x_{l-1})$
4:     $\hat{p} \leftarrow \hat{p} \cdot \frac{q_{l+1}^*(x_l)}{q_l^*(x_l)}$
5: end for
6: return $\hat{p}/Z_L$
```

Figure 4: Pseudocode for generating an unbiased estimate $\hat{p}$ of the probablity of x_0 under a DBN with L layers. Layer l here has unnormalized density q_l^*. Note that in practice, an average over multiple such estimates will generally be taken. Also note that Z_L will typically have to be estimated as well.

where again *const* is constant in the parameters of r. Training the DBN by optimizing the above lower bound therefore minimizes the KL-divergence between the true posterior and the approximating posterior, thereby making it a better proposal distribution. We will further address the question of statistical efficiency empirically in the experimental section.

The definition of the estimator for two-layer DBNs readily extends to DBNs with L layers. If $p(x)$ is the marginal density of a DBN whose layers are RBMs with densities $q_1(x_0, x_1), \ldots, q_L(x_{L-1}, x_L)$ and partition functions $Z_1, \ldots, Z_L$, and if we refer to the states of the random vectors in each layer by $x_0, \ldots, x_L$, where x_0 contains the visible states and x_L contains the states of the top hidden layer, then

$$\begin{aligned} p(x_0) &= \sum_{x_1,\ldots,x_L} q_L(x_{L-1}, x_L) \prod_{l=1}^{L-1} q_l(x_{l-1} \mid x_l) \\ &= \sum_{x_1,\ldots,x_{L-1}} q_L(x_{L-1}) \prod_{l=1}^{L-1} q_l(x_l \mid x_{l-1}) \frac{q_l^*(x_{l-1})}{q_l^*(x_l)} \\ &= q_1^*(x_0) \frac{1}{Z_L} \sum_{x_1,\ldots,x_{L-1}} \prod_{l=1}^{L-1} q_l(x_l \mid x_{l-1}) \frac{q_{l+1}^*(x_l)}{q_l^*(x_l)}. \end{aligned}$$

In order to estimate this term, hidden states $x_1, \ldots, x_{L-1}$ are generated in a feed-forward manner using the conditional distributions $q_l(x_l \mid x_{l-1})$. The weights $\frac{q_{l+1}^*(x_l)}{q_l^*(x_l)}$ are computed along the way, then multiplied together and finally averaged over all drawn states. Intuitively, the estimation process can be imagined as first assigning a basic value using the first layer and then correcting this value based on how the densities of each pair of consecutive layers relate to each other. Pseudocode for this procedure is given in Figure 4.

3.4 Potential Log-Likelihood

In this section, we will discuss a concept which appears in Roux and Bengio (2008) and which we will dub the *potential log-likelihood*. By considering a best-case scenario, the potential log-likelihood can give us an idea of the log-likelihood that can at best be achieved by training additional

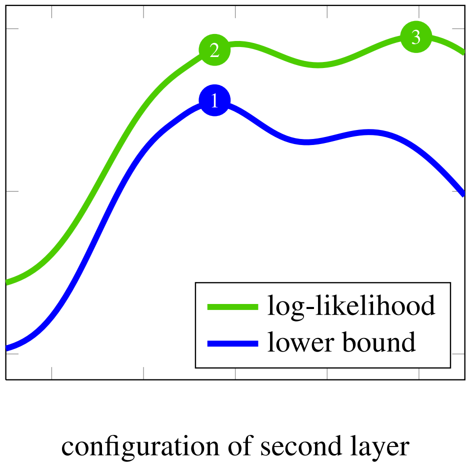

Figure 5: A cartoon explaining the potential log-likelihood. For each choice for the second layer of a DBN, we obtain a different value for the log-likelihood and its lower bound. From all possible choices, we take the distribution which maximizes the lower bound (1) and compute its log-likelihood (2). We call this particular log-likelihood the *potential log-likelihood*. It is still possible that the log-likelihood is larger for other distributions (3). However, it is unlikely that such a distribution will be found, as the second layer is optimized with respect to the lower bound.

layers using the greedy learning of Hinton and Salakhutdinov (2006). Its usefulness will become apparent in the experimental section.

Let $q(x,y)$ be the distribution of an already trained RBM or one of its generalizations, and let $r(y)$ be a second distribution—not necessarily the marginal distribution of any Boltzmann machine. As in section 2.3, $r(y)$ serves to replace the prior distribution over the hidden variables, $q(y)$, and to thereby improve the marginal distribution over x, $\sum_y q(x \mid y)r(y)$. As above, let $\tilde{p}(x)$ denote the data distribution. Our goal is to increase the expected log-likelihood of the model distribution with respect to r,

$$\sum_x \tilde{p}(x) \log \sum_y q(x \mid y) r(y). \tag{12}$$

In applying the greedy learning procedure, we try to reach this goal by optimizing a lower bound on the log-likelihood (4), or equivalently, by minimizing the following KL-divergence:

$$D_{\mathrm{KL}}\left[\sum_x \tilde{p}(x) q(y \mid x) || r(y)\right] = -\sum_x \tilde{p}(x) \sum_y q(y \mid x) \log r(y) + const,$$

where *const* is constant in r.

The KL divergence is minimal if $r(y)$ is equal to

$$\sum_x \tilde{p}(x) q(y \mid x) \tag{13}$$

for every y. Since RBMs are universal approximators (Roux and Bengio, 2008), this distribution could in principle be approximated arbitrarily well by a single, potentially very large RBM. Assume therefore that we have found this distribution, that is, we have maximized the lower bound with respect to all possible distributions r. Then, the distribution for the DBN which we obtain by replacing r in (12) with (13) is given by

$$\begin{aligned}\sum_y q(x \mid y) \sum_{x_0} \tilde{p}(x_0) q(y \mid x_0) &= \sum_{x_0} \tilde{p}(x_0) \sum_y q(x \mid y) q(y \mid x_0) \\ &= \sum_{x_0} \tilde{p}(x_0) q_0(x \mid x_0),\end{aligned}$$

where we have used the *reconstruction distribution*

$$q_0(x \mid x_0) = \sum_y q(x \mid y) q(y \mid x_0),$$

which can be sampled from by conditionally sampling a state for the hidden units, and then, given the state of the hidden units, conditionally sampling a *reconstruction* of the visible units. The log-likelihood we achieve with this lower-bound optimal distribution is given by

$$\sum_x \tilde{p}(x) \log \sum_{x_0} \tilde{p}(x_0) q_0(x \mid x_0).$$

We will refer to this log-likelihood as the *potential log-likelihood*. Note that the potential log-likelihood is not a true upper bound on the log-likelihood that can be achieved with greedy learning, as suboptimal solutions with respect to the lower bound might still give rise to higher log-likelihoods. However, if such a solution was found, it would have been rather by accident than by design. The situation is depicted in the cartoon in Figure 5.

4. Experiments

We performed two types of experiments. In the first, we tested the performance of our estimator. In the second, we further investigated the capabilities of a DBN architecture proposed by Osindero and Hinton (2008) as a model for natural images. The model consists of a GRBM in the first layer and SRBMs in the subsequent layers.

For all but the topmost layer, our estimator requires evaluation of the unnormalized marginal density over hidden states (see Figure 4). In an SRBM, however, integrating out the visible states is analytically intractable. Fortunately, an RBM with the same hidden-to-visible connections and the same bias weights but without the lateral connections turned out to be an efficient enough proposal distribution. We estimated the unnormalized SRBM marginals using

$$q^*(y) = \sum_x q^*(x,y) = \sum_x q'(x \mid y) \frac{q^*(x,y)}{q'(x \mid y)} \approx \frac{1}{N} \sum_n \frac{q^*(x^{(n)},y)}{q'(x^{(n)} \mid y)},$$

where q' is the density of an RBM which approximates the density of the SRBM, q.

4.1 Testing the Estimator

As a first test, we considered a three-layer model for which the likelihood is still tractable. It employed 15 hidden units in each of the first two layers, 50 hidden units in the third layer and was

layers	true neg. log-likelihood	est. neg. log-likelihood
1	2.0482569	2.0484440
2	2.0478912	2.0477577
3	2.0478672	2.0474685

Table 1: True and estimated negative log-likelihood in bits per component of a small DBN trained on 4 by 4 image patches. The estimated likelihood closely matches the true likelihood. Adding more layers to the network did not help to improve the performance if the GRBM employed only few hidden units.

trained on 4 by 4 pixel image patches taken from the van Hateren image data set (van Hateren and van der Schaaf, 1998). The image patches were preprocessed using a standard battery of preprocessing steps including a log-transformation, a centering step and a whitening step. Additionally, the DC component was projected out and only the remaining 15 components of each patch were used for training (for a more detailed description of the preprocessing, see Eichhorn et al., 2009). Brute-force and estimated results are given in Table 1.

We also compared results obtained with our estimator applied to larger models to results obtained by Murray and Salakhutdinov (2009). Using the same preprocessing of the image patches and the same hyperparameters as in Murray and Salakhutdinov (2009), we trained a two-layer model with 2000 hidden units in the first layer and 500 hidden units in the second layer on 20 by 20 pixel van Hateren image patches. In contrast to the preprocessing of the smaller image patches, the DC component was not removed. We used 150000 patches for training and 50000 patches for testing and obtained a negative log-likelihood of 2.047 bits per component on the test set, compared to 2.032 bits reported by Murray and Salakhutdinov (2009).

We further evaluated a two-layer model with 500 hidden units in the first and 2000 hidden units in the second layer trained on a binarized version of the MNIST data set. For the first layer we took the parameters from Salakhutdinov (2009), which were available online. We then tried to match the training of the second-layer RBM with 2000 hidden units using the information given in Salakhutdinov (2009) and Murray and Salakhutdinov (2009). Evaluating the model on the whole test set, we obtained a result of 0.357 bits per component compared to 0.358 bits reported by Murray and Salakhutdinov (2009).

We measured the statistical efficiency of our estimator in terms of the *effective sample size* (ESS) (Kong et al., 1994). For N samples, target density p and proposal density q, the ESS is defined to be

$$\frac{N}{1+\mathbf{Var}_q(y)\left[\frac{p(y)}{q(y)}\right]} = \frac{N}{1+D_2[p(y)||q(y)]}.$$

D_2 is the α-divergence with $\alpha = 2$ and is also known as the χ^2-divergence. In our case, the target distribution is the conditional distribution over the hidden states of a DBN given a state for the visible units. If p equals q, then the variance of the importance weights is zero, so that a single sample from the posterior would suffice to get a perfect estimate of the unnormalized probability of any data point. In that case, the ESS would be N.

Note that the success of training DBNs is also influenced by the ESS. The tightness of the lower bound optimized during training depends on the goodness of the same approximation, although it is measured in terms of the KL-divergence rather than the χ^2-divergence (see Equation 11).

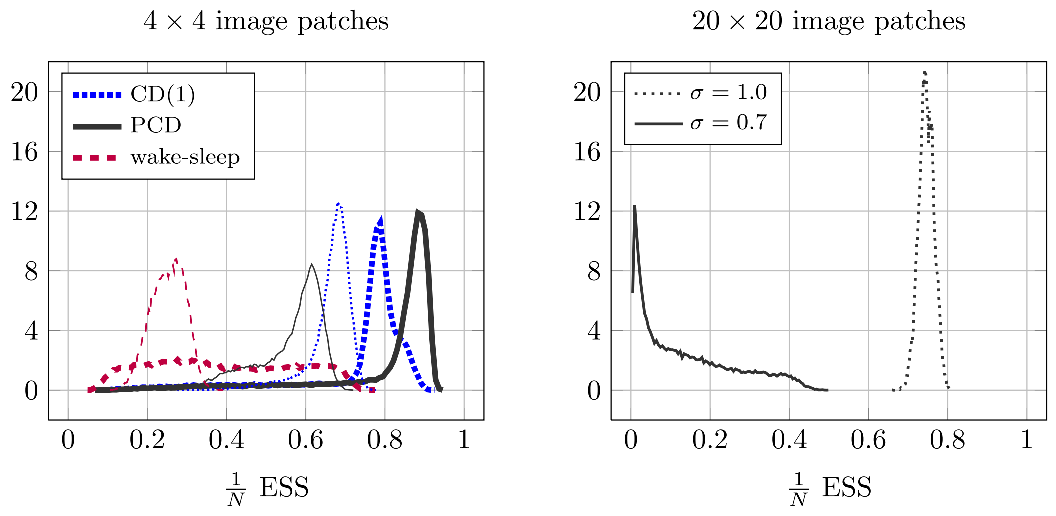

Figure 6: *Left:* Distributions of relative effective sample sizes (ESS) for two-layer DBNs (thick lines) and three-layer DBNs (thin lines) trained on 4 by 4 pixel image patches using different training rules. The larger the ESS for a given data point, the more efficient is our estimator in determining its probability. *Right:* Ditributions of relative ESSs for two two-layer models trained on 20 by 20 pixel image patches. Both models were trained with PCD but using different hyperparameters. While the GRBM yielded a better performance for smaller σ, it also caused the evaluation and the training of subsequent layers to be more difficult.

Because the distributions we are interested in are conditional distributions which are dependent on the state of the visible units, we obtain a different ESS for each data point. Hence, we get a distribution over ESSs for each model (Figure 6). We found that the quality of the proposal distribution was strongly dependent on the parameters of the model. Training a model on the larger image patches using the same hyperparameters as used by Osindero and Hinton (2008) and Murray and Salakhutdinov (2009), for example, led to a distribution of ESSs which was sharply peaked around 0.75. This made it easier to train subsequent layers and to evaluate the two-layer model. Using a smaller value for the Gaussian noise, on the other hand, led to a better likelihood but also a worse approximation to the posterior distribution. Consequently, getting accurate estimates of the log-likelihood took longer (right plot in Figure 7).

On a single core of an AMD Opteron 6174 machine with 2.20 GHz and with an implementation written in Python, it took us about 10 minutes to get accurate results for 50 data points taken from the MNIST data set. Murray and Salakhutdinov (2009) report that they needed about 50 minutes on a Pentium Xeon 3.00 GHz to get stable results for 50 data points. However, a fair comparison of computational efficiency is difficult, even when the comparison is performed with the same machine and the same programming framework is used. For the larger models trained on 20 by 20 van Hateren image patches, it took us less than a second and up to a few minutes to get reasonably accurate estimates for 50 samples (Figure 7). Note that even for the case where the distribution of ESSs is peaked around a value very close to zero, evaluating a large number of training samples was

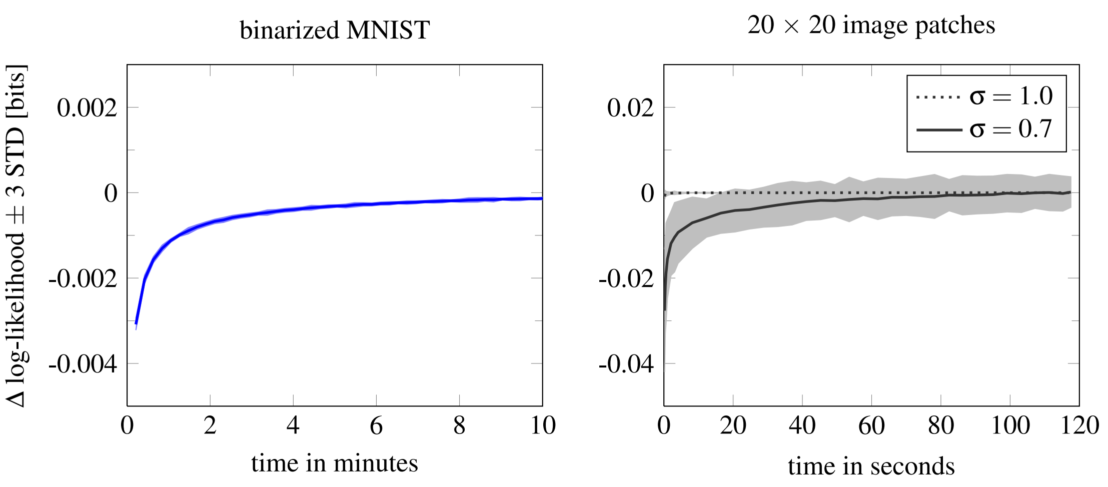

Figure 7: *Left:* The log-likelihood in bits per component was estimated for 50 test samples using different numbers of proposal samples. The difference between each estimate and the most accurate estimate was then measured. The plots show the mean difference and its standard deviation relative to the time it takes to compute the estimates. The bias in the estimates is due to the logarithm. *Right:* The same plot for a different model. The time it takes to get accurate estimates depends on the parameters of the model. Using the parameters obtained with the hyperparameters suggested in (Osindero and Hinton, 2008), already a single sample yielded good estimates of the log-likelihood (about 0.2 seconds). But even for the case where our estimator showed the least statistical efficiency, it only took a few minutes to evaluate 50 samples on a single CPU.

more than feasible. To further reduce computation time, the evaluation can easily be parallelized by splitting the test set into batches which are evaluated separately.

One likely reason for the efficiency of our estimator is the optimization of the lower bound during training (see discussion around Equation 11). This means that if the model is trained with a different learning algorithm, it should become less efficient. We found that this was indeed the case if models were further optimized using the wake-sleep algorithm (Figure 6).

4.2 Model Comparison

We compared the DBN's performance to the performance of complete ICA (Comon, 1994; Eichhorn et al., 2009) as well as several mixture distributions. Perhaps closest in interpretation to the GRBM as well as to the DBN is the mixture of isotropic Gaussian distributions (MoIG) with identical covariances and varying means. Recall that the GRBM can be interpreted as a mixture of a very large number of isotropic Gaussian distributions with weight sharing constraints. After the parameters of the GRBM have been fixed, adding layers to the network only changes the prior distribution over the Gaussians, but does not alter their means. Other models taken into account are mixtures of Gaussians with unconstrained covariance but zero mean (MoG) and mixtures of elliptically contoured distributions with zero mean (MoEC) (Bethge and Hosseini, 2007), of which a special case is the more well-known mixture of Gaussian scale mixtures (e.g., Guerrero-Colon et al., 2008).

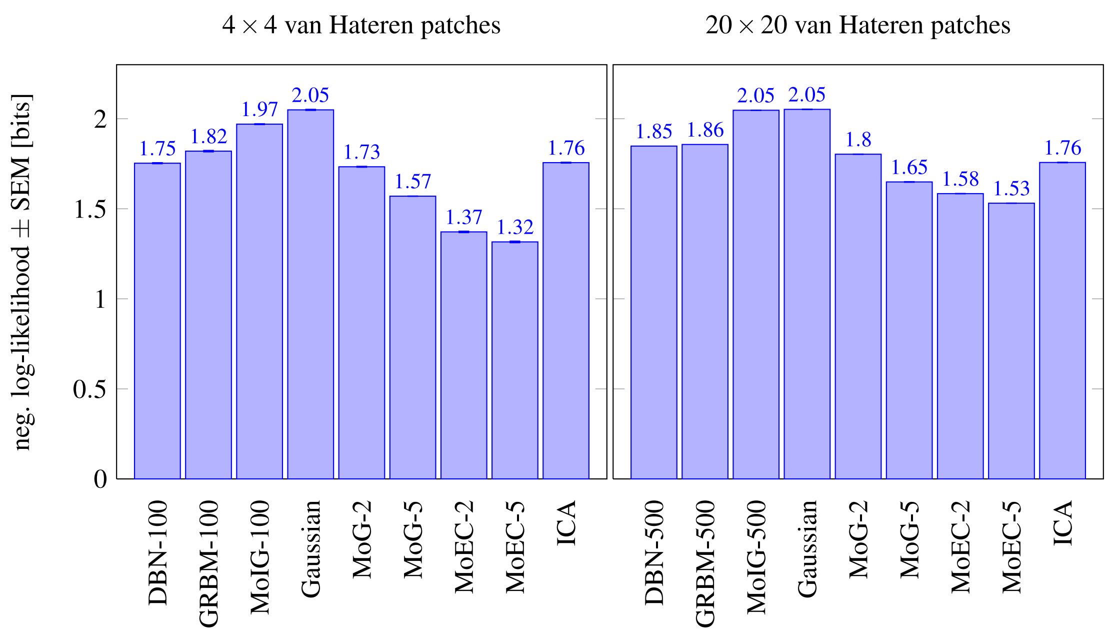

Figure 8: A comparison of different models. For each model, the estimated negative log-likelihood in bits per data component is shown, averaged over 10 independent trials with independent training and test sets. For the GRBM and DBN trained on 20 by 20 image patches, the performance obtained from a single trial is shown. The number behind each model hints either at the number of hidden units or at the number of mixture components used. DBNs were trained using the greedy learning rule and fine-tuned using the wake-sleep algorithm. Boltzmann machines were trained with PCD. Larger values correspond to worse performance.

Using PCD, we trained a three-layer model with 100 hidden units per layer on the smaller 4 by 4 image patches. As expected, both the GRBM and the DBN performed better than the mixture of isotropic Gaussians. Strikingly, however, the DBN was outperformed even by the mixture of Gaussians with just two components. The overall results suggest that mixture models with freely varying covariance are better suited for modeling the statistics of natural images than mixture models with constrained covariance (Figure 8), which is consistent with previous observations that having a flexible model of the covariance structure is important (e.g., Karklin and Lewicki, 2009; Ranzato et al., 2010a).

Adding a second layer to the network only helped very little and we found that the better the performance achieved by the GRBM, the smaller the improvement contributed by subsequent layers. For the hyperparameters that led to the best performance, adding a third layer had no effect on the likelihood and in some cases even led to a decrease in performance (Figure 9). This was despite the apparently reasonable approximation to the posterior distribution over the hidden states (Figure 6). The hyperparameters were selected by performing separate grid searches for the different layers (for details, see Appendix A). PCD on average yielded a slightly worse performance than CD(5) or CD(10), indicating that the Markov chain used during training to sample from the model converged

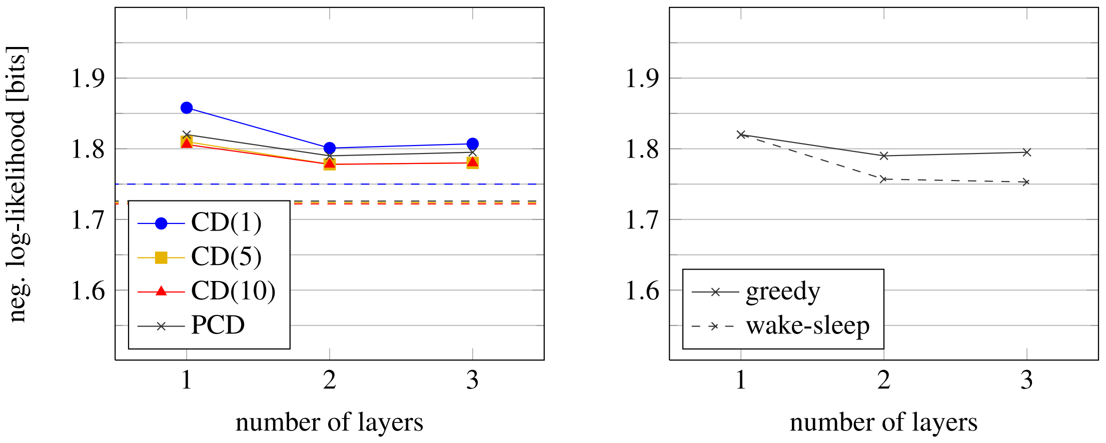

Figure 9: *Left:* Estimated performance of three DBN-100 models trained on 4×4 van Hateren image patches using the greedy learning rule. Each point represents an average over 10 models trained on different training sets. Smaller values correspond to better performance. Error bars were too small to be visible and were therefore left out. The dashed lines indicate the estimated negative potential log-likelihood. *Right:* Performance after fine-tuning the model parameters with the two-layer and three-layer models with the wake-sleep algorithm.

quickly to the true distribution. For other sets of hyperparameters we found that adding a third layer was still able to yield some improvement, but the overall performance of the three-layer model achieved with these hyperparameters was worse.

We also trained a two-layer DBN with 500 hidden units in the first layer and 2000 hidden units in the third layer on 20 by 20 pixel image patches. Using PCD instead of CD(1) and a smaller value for σ led to a performance which is about 0.18 bits per pixel better than the performance reported by Murray and Salakhutdinov (2009). However, the performance achieved by the model was again worse than the performance achieved by other, simpler models. As for the smaller model, the improvement in performance of the GRBM led to a smaller improvement gained by adding a second layer to the model. Despite the much larger second layer, the performance gain induced by the second layer was even less than for the smaller models. Both patch sizes led to the same ordering of the models and resulted in an overall similar picture (Figure 8).

The improvement of the DBN over the GRBM trained with PCD is about 0.05 bits per component for the smaller model and 0.01 bits for the larger model. An important question is why this improvement is so small. Insight into this question can be gained by evaluating the potential log-likelihood, as it represents a practical limit to the performance which can be achieved by means of the approximate greedy learning procedure and could in principle be evaluated even before training any additional layers. If the potential log-likelihood of a GRBM is close to its log-likelihood, adding layers is a priori unlikely to prove useful. Unfortunately, exact evaluation of the potential log-likelihood is intractable, as it involves two nested integrals with respect to the data distribution,

$$\int \tilde{p}(x) \log \int \tilde{p}(x_0) q_0(x \mid x_0) dx_0 dx. \tag{14}$$

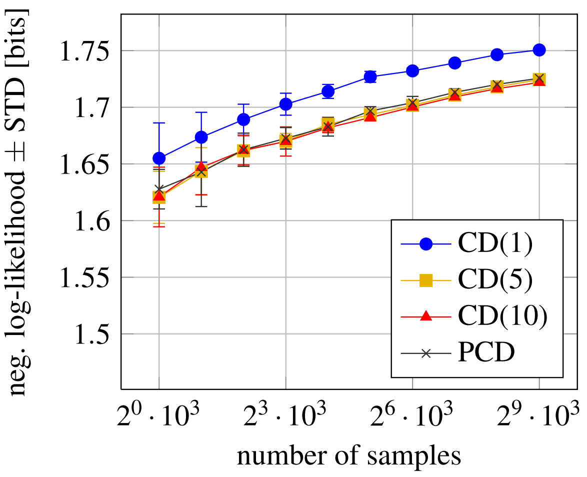

Figure 10: Estimated negative potential log-likelihood of the GRBM. Each plot represents an average over 10 GRBMs trained on different training sets. Error bars indicate one standard deviation. After 512000 samples to approximate the integrals of the potential log-likelihood, the estimates have still not converged, suggesting that the true potential log-likelihood is even worse.

Nevertheless, optimistic estimates of this quantity can still tell us something about the DBN's capability to improve over the GRBM. We estimated the potential log-likelihood using the same set of data samples to approximate both integrals, thereby encouraging optimistic estimation. Note that estimating the potential log-likelihood in this manner is similar to evaluating the log-likelihood of a kernel density estimate on the training data, although the reconstruction distribution $q_0(x \mid x_0)$ might not correspond to a valid kernel. Also note that by taking more and more data samples, the estimate of the potential log-likelihood should become more and more accurate. Figure 10 therefore suggests that the negative potential log-likelihood of a GRBM trained with PCD is at least 1.72 or larger, which is still worse than the performance of, for example, the mixture of Gaussian distributions with 5 components.

The potential log-likelihood is a joint property of the trained first-layer model and the greedy learning procedure. If the greedy algorithm is responsible for the weak performance of the model, it should be possible to get better results by using a non-greedy strategy. We therefore tried to improve the model's performance by fine-tuning the weights with the wake-sleep algorithm (Hinton et al., 1995). We trained the model for another 100 epochs using the same set of hyperparameters but with smaller learning rates. This led to a small improvement (Figure 9), but the likelihood remained behind the performance of the mixture of Gaussians with two components. By applying the same strategy to the larger model, we were unable to get any further improvements. If the learning rate was chosen large enough to cause a measurable change in performance, the performance of the model decreased.

5. Discussion

In this paper, we have introduced a new estimator for the likelihood of DBNs. We have shown that it can be very efficient for models trained with the greedy learning algorithm and efficient enough to evaluate models fine-tuned using the wake-sleep algorithm. However, as we have also seen, the estimator's performance depends on the particular parameters of the model and the way the model was trained. The estimator is unbiased for the unnormalized likelihood, but only asymptotically unbiased for the log-likelihood. When evaluating the log-likelihood, this means that the number of samples from the proposal distribution has to be chosen large enough to ensure that the effects of the bias are small. Since the number of proposal samples is the only parameter of our estimator and the evaluation can generally be performed quickly, a good way to do this is to test the estimator for different numbers of proposal samples on a smaller test set.

Using our estimator, we have shown that DBNs based on GRBMs and SRBMs are not well suited for capturing the statistics of natural images. Since the family of DBNs includes a very large number of models if we allow arbitrary Boltzmann machines in the definition, this of course does not imply that no DBN is able to model natural images well. In fact, other models like the mcRBM or mPoT (Ranzato and Hinton, 2010; Ranzato et al., 2010b) promise to be much better models of natural images than the GRBM and can be used to construct DBNs. One goal of this paper, however, was to see if a deep network with simple layer modules could perform well as a generative model for natural images. While more complex Boltzmann machines are likely to achieve a better likelihood, it is not clear why they should also be good choices as layer modules for constructing DBNs. In particular, it is not clear why they should also lead to large improvements through the addition of layers. In our experiments, better performances achieved with the first layer were always accompanied by a decrease in the performance gained by adding layers. Note that a model which achieves a high likelihood could still have a potential log-likelihood very close to it. This would make a model a better model for natural images, but not the best choice as a layer module for DBNs trained with the greedy learning algorithm.

In Ranzato et al. (2011) it was shown that adding a second-layer RBM to the mPoT model leads to qualitatively different and visually more appealing samples. However, as we have shown in this work, the appearance of samples can be misleading when judging the generative performance of a probabilistic model. For the DBN based on GRBMs and SRBMs we arrive at a different conclusion about the model's capabilities than Osindero and Hinton (2008), who based their conclusions mainly on statistics derived from model samples. We believe that it is therefore worth the effort to come up with statistical estimators which directly target the likelihood or related quantities.

The relationship between generative models and models used for object recognition is not yet fully understood. Hence, the implications of our results on the object recognition performance of features learned by a DBN are also not clear. Most DBNs used to learn features for solving supervised tasks rarely exceed more than a few layers (e.g., Ranzato and Hinton, 2010 show that the effects of adding layers to a GRBM or mcRBM on object recognition performance can be small or even adverse). This observation might be related to the observations made here, that adding layers does not help much to improve the likelihood on natural image patches. A possible explanation for the small contribution of each layer would be that too little information is carried by the hidden representations of each layer about the respective visible states. This is consistent with a small potential log-likelihood and would affect both object recognition and generative performance. If the visible states can be perfectly reconstructed from the hidden states, that is, when $q_0(x \mid x_0) =$

$\delta(x - x_0)$, the potential log-likelihood reduces to the negative entropy of the data—the maximum value for the likelihood that any model can achieve.

A much less frequently used hierarchical model is *hierarchical ICA*, for which a greedy learning algorithm was introduced by Chen and Gopinath (2001) and which can be seen as a special case of projection pursuit density estimation. In ICA, all the information about a visible state is retained in the hidden representation, so that the potential log-likelihood is optimal for this layer module. As shown in Figure 8, already a single ICA layer can compete with a DBN based on RBMs. Furthermore, adding layers to the network is guaranteed to improve the likelihood of the model (Chen and Gopinath, 2001) and not just a lower bound as with the greedy learning algorithm for DBNs. Hosseini and Bethge (2009) have shown that adding layers does indeed give significant improvements of the likelihood, although it was also shown that hierarchical ICA cannot compete with other, non-hierarchical models of natural images. An interesting question is whether representations learned by this model can also compete or outperform those learned by DBNs in supervised tasks. Comparing the supervised performance of features learned with hierarchical ICA and the model discussed here could shed further light on the importance of the likelihood when training DBNs for supervised tasks.

A lot of research has been devoted to creating new layer modules (e.g., Roux et al., 2010; Ranzato and Hinton, 2010; Lee and Ng, 2007; Welling et al., 2005) and finding better approximations to ML learning for training these (e.g., Gutmann and Hyvärinen, 2010; Sohl-Dickstein et al., 2009; Tieleman, 2008). To our knowledge, much less work has been done to improve the general strategy for training DBNs since its introduction. Currently, the lower layers are trained in a way which is independent of whether additional layers will be added to the network. A better performance could be achieved by devising alternative learning schemes in which the lower layers are optimized to better assist the upper layers, for example by making sure that the potential log-likelihood stays large. Directly regularizing with respect to the potential log-likelihood is difficult due to the nested integrals (see Equation 14). An improvement could nevertheless indirectly be achieved by maximizing the information that is carried in the hidden representations about the states of the visible units.

An alternative strategy would be to use non-greedy learning strategies such as the wake-sleep algorithm, for which the potential log-likelihood no longer plays a critical role. We showed that in one case, the wake-sleep algorithm was able to further improve the likelihood. In another case, we were unable to get any improvement. While more extensive tests are necessary before final conclusions about its effectiveness should be made, its potential to improve the likelihood of the DBN discussed here seems rather limited.

A very recent paper by Ngiam et al. (2011) introduced an alternative approach to hierarchical modeling of natural images. Instead of stacking probabilistic models on top of each other, the approach is based on a single Boltzmann machine in which the energy function itself is hierarchically organized. This allows them to jointly train all parameters of the network and also to evaluate the likelihood much more easily. The paper reports the likelihood of several instances of the model and shows that the model is able to take advantage of multiple layers. Unfortunately, it missed to also report the likelihood of other, more well known models of natural images. This makes it hard to judge the performance of the model, as the absolute value of the likelihood is highly dependent on the preprocessing of the data.

Despite good arguments for why hierarchical models should excel at modeling natural images, no hierarchical model has been convincingly shown to yield state-of-the-art generative performance and outperform other, much simpler models, such as the mixture of Gaussians, in terms of the log-likelihood.

Acknowledgments

We would like to thank the anonymous reviewers for helpful comments and suggestions. We also thank Iain Murray for providing us with a binarized version of the MNIST data set. This work is supported by the German Ministry of Education, Science, Research and Technology through the Bernstein award to Matthias Bethge (BMBF, FKZ: 01GQ0601) and the Max Planck Society.

Appendix A.

In all experiments, we used stochastic gradient descent with a batch size of 100 data points to train DBNs. For the 4 by 4 pixel image patches, we used $5 \cdot 10^4$ training and $5 \cdot 10^4$ test samples in each trial. For the case of 20 by 20 pixel image patches, we used $15 \cdot 10^4$ training and $5 \cdot 10^4$ test samples.

When PCD was used for training, the persistent Markov chain was updated using one Gibbs sampling step before each computation of the gradient.

For the GRBM with 100 hidden units trained on 4 by 4 pixel image patches, we used a σ of 0.65 and trained the models for 200 epochs. Together with CD(1) we used a learning rate of 10^{-2} and weight decay of 10^{-2} times the learning rate. With PCD, CD(5) and CD(10) we used a learning rate of $5 \cdot 10^{-3}$. For all parameters and all models, a momentum parameter of 0.9 used.

We initialized the second-layer SRBM so that its marginal distribution matched that of the GRBM and trained it for 200 epochs. During training, approximate samples from the conditional distribution of the visible units were obtained using 20 parallel mean field updates with a damping parameter of 0.2 (Welling and Hinton, 2002). For all sampling schemes, we used a learning rate of $5 \cdot 10^{-3}$ for hidden-to-visible connections and a learning rate of $5 \cdot 10^{-4}$ for lateral connections. The weight decay was set to $5 \cdot 10^{-3}$. The third layer was randomly initialized and had to be trained for 500 epochs before convergence was reached. It employed the same hyperparameters as the second layer.

The hyperparameters were selected by performing several grid searches. After the hyperparameters of the GRBM had been selected, we performed a second grid-search for the second-layer SRBM and used the same hyperparameters for the third-layer SRBM. We performed separate grid searches for CD(1) and PCD, but used the hyperparameters found for PCD also with CD(5) and CD(10). Each grid search comprised 75 hyperparameter combinations for the GRBM and 45 hyperparameter combinations for the SRBM. After the hyperparameters had been selected, we retrained the model.

For the larger model applied to 20 by 20 pixel image patches, we trained both layers for 200 epochs using PCD. For the GRBM, we used a σ of 0.7, a learning rate of 10^{-3} and weight decay of 10^{-2}. For the SRBM, we used a learning rate of 10^{-3} for the hidden-to-visible connections and $5 \cdot 10^{-4}$ for the lateral connections. We used 30 parallel mean field updates with a damping rate of 0.2 to sample the visible units. The hyperparameters of the GRBM were selected by performing a grid search over 66 hyperparameter combinations. For the SRBM we tried 18 different combinations. Due to the high computational cost of training the two-layer model, we took the parameters which achieved the best result in the grid search for our comparison and evaluated it using a separate test set.

Partition functions were estimated using AIS. For the smaller model we used 10^3 intermediate distributions with a linear annealing schedule, 100 samples in the first and 10^3 samples in the second and third layer. For the larger model we used a linear annealing schedule, $2 \cdot 10^4$ intermediate distributions and 100 importance samples in both layers. The transition operator for the SRBM was implemented using Gibbs sampling. Conditioned on the hidden units, the visible units were updated in a random order.

To estimate the unnormalized log-probability of each data point with respect to the smaller models, we used 200 samples from the proposal distribution of our estimator for the two-layer, and 500 samples for the three-layer model. For the larger model we used 2000 samples, although less would have been sufficient.

For the fine-tuning of the smaller models with the wake-sleep algorithm, we reduced the learning rates to $\frac{1}{4}$ of the original learning rates. Using larger learning rates led to a decrease in performance. We used the same hyperparameters for the DBN representing the proposal distribution. We trained the models for 100 epochs. Training the models for 200 epochs led to no further improvements. We controlled for the estimator's bias by testing the estimators behavior on a smaller test set for different numbers of proposal samples. We used 4000 samples from the proposal distribution of our estimator to estimate the log-likelihood. Using only 500 samples led to essentially the same results, but the small difference in performance between the two-layer and three-layer model was less visible.

To estimate the unnormalized hidden marginals of SRBMs, we used 100 proposal samples from an approximating RBM with the same hidden-to-visible connections and same bias weights.

Lastly, note that the performance of the GRBM and the DBN might still be improved by taking a larger number of hidden units. A post-hoc analysis revealed that the GRBM does indeed not overfit but continues to improve its performance if the variance is decreased while increasing the number of hidden units. This is of course also true for the other models we evaluated, whose performance can still be improved if we allow them to use more parameters.

Code for training and evaluating deep belief networks using our estimator can be found under

`http://www.bethgelab.org/code/theis2011/`.

References

M. Bethge and R. Hosseini. Method and device for image compression. Patent WO/2009/146933, 2007.

S. S. Chen and R. A. Gopinath. Gaussianization. *Advances in Neural Information Processing Systems 13*, 2001.

P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

J. Eichhorn, F. Sinz, and M. Bethge. Natural image coding in V1: How much use is orientation selectivity? *PLoS Computational Biology*, 5(4), 2009.

D. J. Felleman and D. C. van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1991.

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 1980.

J. A. Guerrero-Colon, E. P. Simoncelli, and J. Portilla. Image denoising using mixtures of gaussian scale mixtures. *Proceedings of the 15th IEEE International Conference on Image Processing*, 2008.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, Jan 2006.

G. E. Hinton, P. Dayan, B. Frey, and R. Neal. The ”wake-sleep” algorithm for unsupervised neural networks. *Science*, Jan 1995.

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, Jul 2006.

R. Hosseini and M. Bethge. Hierachical models of natural images. *Frontiers in Computational Neuroscience*, 2009.

Y. Karklin and M. S. Lewicki. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*, 2009.

A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

H. Lee and A. Ng. Sparse deep belief net model for visual area v2. *Advances in Neural Information Processing Systems 19*, 2007.

P. Long and R. Servedio. Restricted boltzmann machines are hard to approximately evaluate or simulate. *Proceedings of the 27th International Conference on Machine Learning*, 2010.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

T. Minka. Divergence measures and message passing. *Microsoft Research Technical Report (MSR-TR-2005-173)*, 2005.

A. Mohamed, G. Dahl, and G. E. Hinton. Deep belief networks for phone recognition. *NIPS 22 workshop on deep learning for speech recognition*, 2009.

I. Murray and R. Salakhutdinov. Evaluating probabilities under high-dimensional latent variable models. *Advances in Neural Information Processing Systems 21*, 2009.

R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, Jan 2001.

J. Ngiam, Z. Chen, P. Koh, and A. Y. Ng. Learning deep energy models. *Proceedings of the 28th International Conference on Machine Learning*, 2011.

S. Osindero and G. E. Hinton. Modeling image patches with a directed hierarchy of markov random fields. *Advances in Neural Information Processing Systems 20*, 2008.

M. Ranzato and G. E. Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, May 2010.

M. Ranzato, A. Krizhevsky, and G. E. Hinton. Factored 3-way restricted boltzmann machines for modeling natural images. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010a.

M. A. Ranzato, V. Mnih, and G. E. Hinton. Generating more realistic images using gated mrfs. *Advances in Neural Information Processing Systems 23*, 2010b.

M. A. Ranzato, J. Susskind, V. Mnih, and G. E. Hinton. On deep generative models with applications to recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

N. Le Roux and Y. Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

N. Le Roux, N. Heess, J. Shotton, and J. Winn. Learning a generative model of images by factoring appearance and shape. *Microsoft Research Technical Report (MSR-TR-2010-7)*, 2010.

R. Salakhutdinov. *Learning Deep Generative Models*. PhD thesis, Dept. of Computer Science, University of Toronto, Sep 2009.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. *Proceedings of the 25th International Conference on Machine Learning*, 25, Apr 2008.

O. G. Selfridge. Pandemonium: A paradigm for learning. *Mechanisation of thought processes: Proceedings of a symposium held at the National Physical Laboratory*, pages 115–122, 1958.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1:194–281, Jan 1986.

J. Sohl-Dickstein, P. Battaglino, and M. R. DeWeese. Minimum probability flow learning. *pre-print*, 2009. arXiv:0906.4779.

J. M. Susskind, G. E. Hinton, J. R. Movellan, and A. K. Anderson. Generating facial expressions with deep belief nets. *Affective Computing, Emotion Modelling, Synthesis and Recognition*, pages 421–440, 2008.

I. Sutskever and G. E. Hinton. Deep, narrow sigmoid belief networks are universal approximators. *Neural Computation*, 20(11):2629–2636, Nov 2008.

G. W. Taylor, G. E. Hinton, and S. Roweis. Modeling human motion using binary latent variables. *Advances in Neural Information Processing Systems 19*, 2007.

T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. *Proceedings of the 25th International Conference on Machine Learning*, 2008.

J. H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society B: Biological Sciences*, 265(1394), Mar 1998.

M. Welling and G. E. Hinton. A new learning algorithm for mean field boltzmann machines. *International Joint Conference on Neural Networks*, 2002.

M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. *Advances in Neural Information Processing Systems 17*, 2005.

L. Younes. Parametric inference for imperfectly observed gibbsian fields. *Probability Theory and Related Fields*, 1989.