Deep Boltzmann Machines as Feed-Forward Hierarchies

Grégoire Montavon
Machine Learning Group
TU Berlin

Mikio L. Braun
Machine Learning Group
TU Berlin

Klaus-Robert Müller
Machine Learning Group
TU Berlin

Abstract

The deep Boltzmann machine is a powerful model that extracts the hierarchical structure of observed data. While inference is typically slow due to its undirected nature, we argue that the emerging feature hierarchy is still explicit enough to be traversed in a feed-forward fashion. The claim is corroborated by training a set of deep neural networks on real data and measuring the evolution of the representation layer after layer. The analysis reveals that the deep Boltzmann machine produces a feed-forward hierarchy of increasingly invariant representations that clearly surpasses the layer-wise approach.

1 Introduction

One of the most interesting features of neural networks is their ability to extract meaningful abstractions of data without being explicitly taught to do so (Hinton and Salakhutdinov, 2006; Guyon et al., 2011). Hopefield networks (Hopfield, 1982) and Boltzmann machines (Ackley et al., 1985; Hinton and Sejnowski, 1986) are theoretically able to find solutions that make very efficient use of the statistical information contained in the data. In the general case, the solution learned by these algorithms is unsuited for direct use in a feed-forward classifier as the prediction typically depends on the state of the hidden variables, thus requiring an iterative algorithm to infer them.

A common approach to alleviate the need for iterative inference is to decompose the architecture into a sequence of simple feed-forward submodules that are learned one after the other. Example of practical algorithms include the deep belief network (Hinton et al., 2006) and the stacked auto-encoder (Bengio et al., 2007) where layers are learned one after the other and stacked on top of each other. In this approach, learning is made easier and inference can be performed in a feed-forward manner since the hidden units of each layer can be evaluated immediately from the visible units. Unfortunately, the layer-wise approach is limited by its greedy training procedure and its inability to combine top-down and bottom-up signals, making it hard to build a coherent feature hierarchy.

In this paper, we argue that it is not necessary to build the feed-forward feature hierarchy explicitly as it is done by the layer-wise approach. Instead, the feature hierarchy arises naturally from the structure of the neural network, even if the latter is undirected. By constraining the Boltzmann machine into a hierarchy of multiple layers, the so-called deep Boltzmann machine (Salakhutdinov and Hinton, 2009) learns a hierarchy of increasingly invariant feature extractors that is explicit enough to be traversed in a feed-forward fashion, without requiring to infer the initial state of the hidden layers.

This claim is corroborated by training a deep Boltzmann machine and a deep belief network on real data and comparing the layer-wise evolution of the representation in each of them. More specifically, we measure how the underlying abstract concepts converge progressively towards the leading components of the representation as we integrate more and more layers (Montavon et al., 2011). The analysis reveals that the deep Boltzmann machine outputs a feed-forward hierarchy of increasingly invariant representations that outperforms the more classical layer-wise approach.

2 Background

In this section, we first give some background on the deep Boltzmann machine and the deep belief network. Then, we introduce a method based on kernel PCA to analyze how the representation is formed layer after layer in these deep networks.

In the following lines, the sigmoid function is defined

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX.

as $\text{sigm}(x) = \frac{e^x}{e^x+1}$, the relation $a \sim b$ denotes that the variable a is drawn randomly from a Bernoulli distribution of parameter b and $\langle\cdot\rangle_{P_\mathcal{X}}$ denotes the expectation operator with respect to a probability distribution $P_\mathcal{X}$.

2.1 Deep Boltzmann Machine (DBM)

The deep Boltzmann machine (DBM, Salakhutdinov and Hinton, 2009) is an undirected network of units organized in a layered structure. Its two-layer version is composed of visible units $x \in \{0,1\}^{d_X}$, intermediate hidden units $y \in \{0,1\}^{d_Y}$ and top hidden units $z \in \{0,1\}^{d_Z}$ where (x,y) and (y,z) form two bipartite graphs whose edges are respectively represented by matrices W and V. A DBM is depicted in Figure 1 (left). In order to simplify the equations, we do not write biases. The energy of the network is given by

$$E(x,y,z;\theta) = -y^\top W x - z^\top V y$$

where $\theta = (W,V)$ are the parameters of the system. The following probability is associated to each visible vector x:

$$p(x;\theta) = \frac{\sum_{y,z} \exp(-E(x,y,z;\theta))}{\sum_{\chi,\upsilon,\zeta} \exp(-E(\chi,\upsilon,\zeta;\theta))}$$

From the probability distribution above, the alternate Gibbs sampler

$$\{x \sim \text{sigm}(W^\top y), z \sim \text{sigm}(Vy)\}$$
$$y \sim \text{sigm}(Wx + V^\top z)$$

can be derived. The derivative of the log-likelihood of the observed data with respect to the model parameters takes the simple form

$$\frac{\partial \log p(x)}{\partial W^\top} = \langle xy^\top\rangle_{\text{data}} - \langle xy^\top\rangle_{\text{model}}$$
$$\frac{\partial \log p(x)}{\partial V^\top} = \langle yz^\top\rangle_{\text{data}} - \langle yz^\top\rangle_{\text{model}}$$

where $\langle.\rangle_{\text{data}}$ are the data-dependent statistics obtained by sampling the model conditioned on the visible units clamped to the data and $\langle.\rangle_{\text{model}}$ are the data-independent statistics obtained by sampling freely from the model. The equilibrium is reached when $\langle.\rangle_{\text{model}} \approx \langle.\rangle_{\text{data}}$, that is, when the model approximates the data distribution well. The learning algorithm will tend to lower the energy of the network near the data points and to raise it elsewhere, creating ravines in the energy landscape representing the learned input distribution.

In order to estimate data-dependent statistics, it is common to run a mean-field approximation with

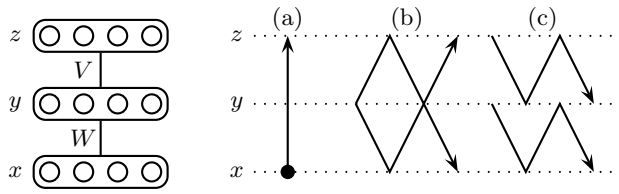

Figure 1: On the left, diagram of a two-layer deep network. On the right, different sampling methods: (a) a feed-forward pass on the network starting from a data point, (b) a single alternate Gibbs sampler on all layers and (c) one alternate Gibbs sampler per layer.

x clamped to the data (Salakhutdinov and Hinton, 2009), that is, computing alternatively

$$z = \text{sigm}(Vy)$$
$$y = \text{sigm}(Wx + V^\top z)$$

until convergence.

On the other hand, in order to estimate data-independent statistics, it is common to run a Markov chain Monte Carlo (Neal, 1993) on the network, that is, running continuously the alternate Gibbs sampler described before on a set of free particles. This approach is illustrated in Figure 1b. This stochastic approximation procedure is typically run in background of the learning algorithm (Tieleman, 2008).

2.2 Feed-Forward DBM (DBM-FF)

We use the term "feed-foward DBM" (or DBM-FF) to denote the simple fact of traversing the trained DBM in a feed-forward manner. The feed-forward pass is depicted in Figure 1a. From a data point x, we compute sequentially $y = \text{sigm}(Wx)$ and $z = \text{sigm}(Vy)$.

2.3 Deep Belief Network (DBN)

The deep belief network (DBN, Hinton et al., 2006) can be seen as a modified version of the DBM where top-down feedback is disabled. The absence of top-down feedback implies that the intermediate units can be inferred directly from the input and that data-dependent statistics can be collected directly by running a feed-forward pass on the network, that is, $x \sim \text{data}$, $y = \text{sigm}(Wx)$ and $z = \text{sigm}(Vy)$. Data-independent statistics are obtained by running a separate Markov chain Monte Carlo on each layer of the network. In the first layer, we compute alternatively $x \sim \text{sigm}(W^\top y)$ and $y \sim \text{sigm}(Wx)$. In the second layer, we compute alternatively $y' \sim \text{sigm}(V^\top z)$ and $z \sim \text{sigm}(Vy')$. This approach is depicted in Figure 1c.

2.4 Layer-Wise Analysis of Deep Networks

We present a method introduced by Montavon et al. (2011) that measures how the representation evolves layer after layer in a deep network. The method is based on the theoretical insight that the projection of the input distribution onto the hidden units of each layer provides a function space that can be thought of as a respective representation and feature extractor.

The method aims to characterize this function space by constructing for each layer a kernel that approximates the implicit transfer function between the input and the layer and measuring how much these kernels "match" the task of interest. The approach is theoretically motivated by the work of Braun et al. (2008) showing that projections on the leading components of the implicit kernel feature map (Schölkopf et al., 1998) obtained with a finite and typically small number of samples n are close with essentially multiplicative errors to their asymptotic counterparts. In the following lines, we describe the principal steps of the analysis:

Let X, T be a data set of n samples where X are the inputs and T are the labels. Let

$$f : x \mapsto f_L \circ \cdots \circ f_1(x)$$

be a deep network made of L layers. We build a hierarchy of increasingly "deep" kernels

$$\begin{aligned} k_{0,\sigma}(x, x') &= \kappa_\sigma(x, x') \\ k_{1,\sigma}(x, x') &= \kappa_\sigma(f_1(x), f_1(x')) \\ &\vdots \\ k_{L,\sigma}(x, x') &= \kappa_\sigma(f_L \circ \cdots \circ f_1(x), f_L \circ \cdots \circ f_1(x')) \end{aligned}$$

that subsume the mapping performed by more and more layers of the deep network and where κ_σ is a RBF kernel of scale σ. For each kernel $k_{l,\sigma}$, we can compute the empirical kernel $K_{l,\sigma}$ of size $n \times n$ and its eigenvectors $u^1_{l,\sigma}, \ldots, u^n_{l,\sigma}$ sorted by decreasing magnitude of their respective eigenvalues. The matrix

$$U^d_{l,\sigma} = (u^1_{l,\sigma} | \ldots | u^d_{l,\sigma})$$

spans the d leading kernel principal components of empirical kernel.

We measure how good a representation is with respect to a certain task by measuring whether the task is contained in the leading principal components of the representation. The prediction and reconstruction error are computed respectively as the residuals of the projection on the d leading kernel principal components:

$$e_T(l, d) = \min_\sigma ||T - U^d_{l,\sigma} U^{d\top}_{l,\sigma} T||^2_F \quad (1)$$

$$e_X(l, d) = \min_\sigma ||X - U^d_{l,\sigma} U^{d\top}_{l,\sigma} X||^2_F \quad (2)$$

Curves $(e(l, 0), \ldots, e(l, d))$ represent how well the task can be solved as we add more and more principal components of the data distribution. These curves can be interpreted as learning curves since the number of observed kernel principal components d closely relates to the amount of label information given to the learning machine. When d is small, we are in the zero-shot learning regime where the model is asked to generalize from very few observations. On the other hand, when d is large, we reach the other extreme case where label information is abundant, and where the representation has to be rich enough in order to encode any subtle variation of the learning problem. These curves can be summarized into one scalar by measuring the area under them:

$$e_T(l) = \frac{1}{n} \sum_{d=1}^{n} e_T(l, d) \quad (3)$$

$$e_X(l) = \frac{1}{n} \sum_{d=1}^{n} e_X(l, d) \quad (4)$$

These aggregated errors are used to compare the layer-wise evolution of the representation in different deep networks.

3 Theoretical Motivations

In this section, we attempt to theoretically motivate the use of deep Boltzmann machines to learn a feed-forward feature hierarchy. The advantages and drawbacks of competing approaches are summarized in Table 1. In particular, we argue that the DBM does a better job than the DBN at organizing the solution into a coherent hierarchy of increasingly invariant representations.

3.1 Building the Invariance

In a deep Boltzmann machine, the learning algorithm encourages the top layer and the input layer to cooperate rather than compete in order to model the data distribution since both layers participate to the reconstruction of the intermediate layer. Indeed, the Gibbs sampler samples the intermediate units as

$$y \sim \text{sigm}(Wx + V^\top z).$$

As a consequence, the top units can specialize on aspects of the underlying data distribution that are complementary to the raw input representation and very importantly that do not necessarily retain all information of the input representation. The fact that top units do not need to retain all information contained in the input is the key factor for building the invariance as the irrelevant pixel variations will typically be filtered out in the feed-forward pass.

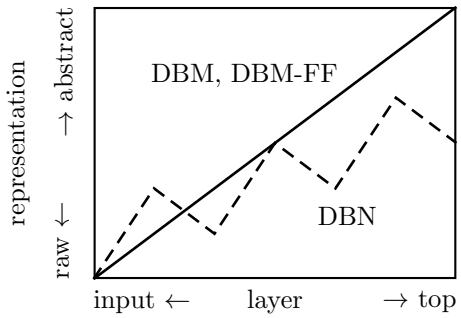

Figure 2: Illustration of the evolution of the representation in a DBM and a DBN as predicted by our theory. The global training procedure of the DBM forces the solution to organize in a hierarchy of increasingly invariant feature extractors while the DBN oscillates layer after layer between abstract and raw representations.

	DBM	DBM-FF	DBN
Feed-forward	No	Yes	Yes
Layer-wise coherence	Yes	Yes	No
Degree of invariance	High	High	Low

Table 1: Characteristics of the three models of interest. The approach advocated in this paper (DBM-FF) combines the speed of feed-forward processing with the ability of deep Boltzmann machines to build a coherent hierarchy of increasingly invariant representations.

On the other hand, the deep belief network is not able to build the same level of invariance as the DBM. Indeed, the input and top layers must independently reconstruct the intermediate layer, which implies that low-level concepts will tend to be propagated to the top of the feature hierarchy instead of being filtered out.

3.2 Preventing Layer-Wise Oscillations

In a deep belief network, layers are agnostic to the abstract representations that could be developed in the upper layers. In order to model the data distribution well, the learning algorithm must therefore build a hidden representation that is the best possible complement of the input representation. Indeed, from the perspective of the Markov chain Monte Carlo, visible and hidden layers cooperate in order to protect themselves from each other and stay confined along the data manifold.

Since layer 0 represents the pixels well, layer 1 will complement the pixels well. When training the second module, since layer 1 complements the pixels well, layer 2 will complement the complement of the pixels well, that is, represent the pixels themselves well. Extrapolating to more layers, an oscillation phenomenon may occur where layers $0, 2, 4, \ldots$ represent the pixels well and layers $1, 3, 5, \ldots$ represent the complement to the pixels well. The effect is depicted in Figure 2 where the representation learned in absence of top-down feedback oscillates layer after layer between raw and abstract representations.

On the other hand, in the deep Boltzmann machine, such oscillations are likely to disappear. Indeed, there is no reason anymore for the top layer to resemble the input layer since the input, intermediate and top layers are all cooperating in order to model the data distribution. From the perspective of the Markov chain Monte Carlo, input and top units are now cooperating in order to protect themselves from intermediate units and stay confined along the data manifold. Input and top units are therefore unlikely to be similar as it would be a waste of capacity.

4 Experiments

In order to verify experimentally the arguments put forward in Section 3, we train a two-layer deep Boltzmann machine and a four-layer deep belief network on data subsets of 2500 samples. The analysis described in Section 2.4 is performed on each trained deep network. In our analysis, we consider the linear kernel and the set of Gaussian kernels with scale corresponding to the 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2 and 0.5 quantiles of the distribution of pairwise distances. The kernel empirical feature map is built using the same 2500 samples used to train the deep network. The small number of samples comes from the hard constraint imposed by the high computational cost of kernel PCA. However, we observe that training a deep network on 2500 samples only is sufficient to highlight the benefits of cross-layer cooperation over the standard layer-wise approach.

For each deep architecture, visible and hidden units are binary. Hidden layers contain 400 units each. The hyperbolic tangent nonlinearity is used instead of the sigmoid. Weights and biases are initialized to zero except for the input bias which is initialized as $b = \tanh^{-1}(\langle x \rangle_{\text{data}})$. The mini-batch and the number of particles of the training algorithm is set to 25, the learning rate is set to $\eta = 10^{-5}$ and the L2 weight penalty is set to $\lambda = 0.04$. We consider the following two data sets (some samples are shown in Figure 3):

- **Handwritten characters recognition:** The data set has been collected by van der Maaten (2009) and consists of 40134 handwritten characters of size 56×56 along with their label (A-Z; 0-9). The task is a 36-class classification task where the label must be predicted from the raw pixel

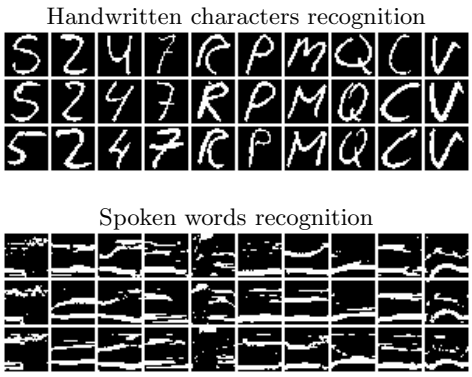

Figure 3: The handwritten characters recognition data set is composed of images of size 28×28 representing characters. The spoken words recognition data set is composed of spectrograms of size 28×28 representing words and where 28 mel-frequency coefficients are taken at 28 evenly distributed time steps.

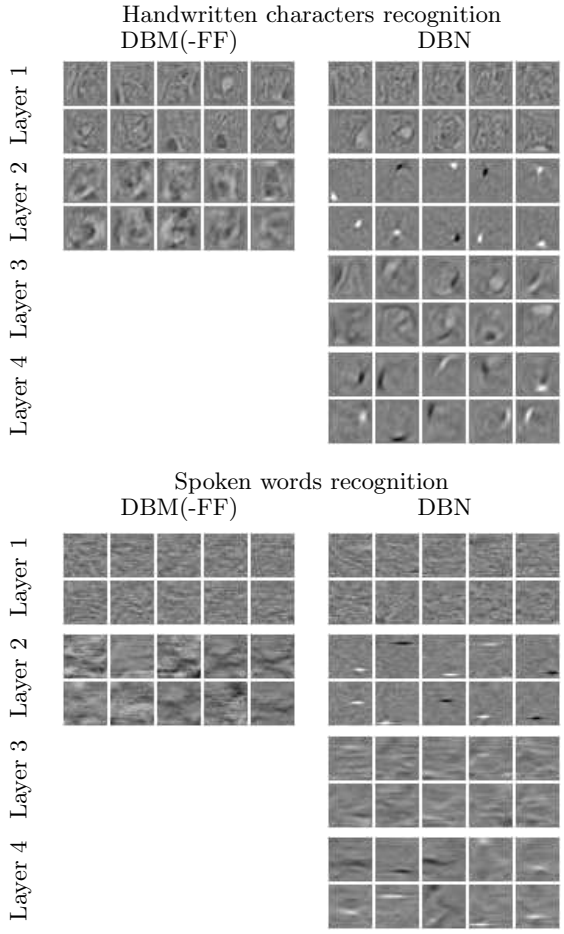

Figure 4: Example of filters learned by the deep networks (visualized by linear projection to the input). In the DBN, filters are alternatively global and local, suggesting that the representation is not evolving monotonically from the pixel to the abstract representation. On the other hand, filters learned by the DBM evolve from local features to global features.

representation. In our experiments, we subsample the characters by a factor two leading to 28×28 binary images. The resulting 784-dimensional signal is fed as input to the deep network.

- **Spoken words recognition:** The data set is based on the Dialect-SA subset of the TIMIT speech corpus[1] consisting of two sentences of respectively 11 and 10 words read by 630 different speakers and labeled with their corresponding phonemes and words. We consider a small vocabulary task that consists of discriminating between the 21 words of the subset, leading to a data set of 13230 samples (21 words $\times$ 630 speakers). Samples are generated by evaluating the spectral power at 28 mel-frequencies and 28 equidistant positions in the word. The mel-frequency spectrum coefficients are obtained by computing the power spectrum on a 20 milliseconds Hamming window and mapping the spectral coefficients on the mel-scale using triangular overlapping windows. The resulting (28×28)-dimensional real-valued spectrum is whitened, binarized and fed as input to the deep network.

We choose these two data sets because they are particularly well suited for unsupervised learning in deep networks. Indeed, handwritten characters or spoken words are taken in a stereotypical pose, that is, centered and isolated, ensuring that the deep network focuses on modeling the object itself and not perturbing elements such as neighboring speech or handwriting.

Also, these objects are believed to be generated by latent variables that represent them more efficiently than the raw input space. These latent variables are moreover related to the pixels through a deep hierarchy of abstractions that can only be modeled efficiently by a hierarchy of multiple layers. For example, a character is composed of strokes that are each composed of pixels. Similarly, a spoken word is composed of syllabes that are composed of phonemes that are composed of formants that are composed of spectral coefficients. In absence of complex deep structures in the data, the deep network is not able to improve the raw representation substantially.

5 Results and Discussion

Results are presented in Figure 5 and 6 and confirm the capacity of the DBM to enable the emergence of a representation that evolves uniformly from the raw sensory input to more abstract concepts as we move from the input to the top layer. Figure 5 shows that

[1] www.ldc.upenn.edu/Catalog/LDC93S1.html

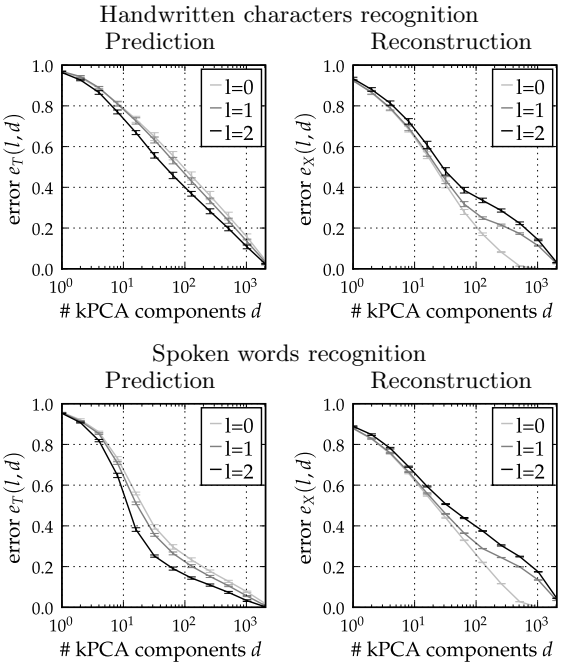

Figure 5: Layer-wise analysis of the DBM-FF (see Eq. 1 and 2). As we move to the top layers, the prediction error decreases and the reconstruction error increases, showing that the labels progressively replace the raw input in the leading components of the representation.

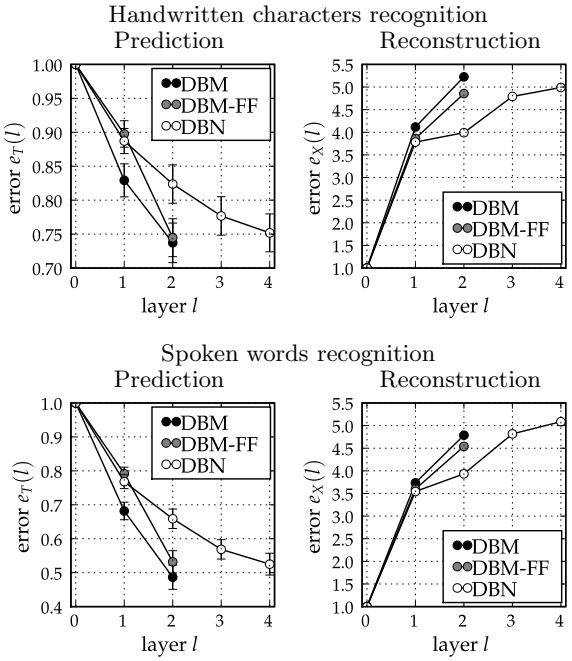

Figure 6: Layer-wise evolution of the representation for each deep network (see Eq. 3 and 4) where curves are normalized so that $e(0) = 1$. The second layer of the DBM is approximated well by a single feed-forward pass. Also, the abstraction is built faster than in a DBN due to the better filtering of irrelevant factors of variation and the absence of layer-wise oscillations.

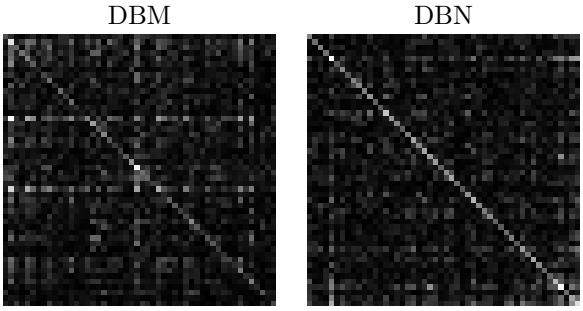

Figure 7: Covariance matrices between top-down and bottom-up contributions on 100 units of the first hidden layer $\langle Wx(V^\top z)^\top \rangle_{\text{data}}$. The diagonal is less marked for the DBM than for the DBN suggesting that the concepts represented in the top layer of the DBM have a certain degree of independence with respect to the concepts represented in the input layer. Such separation of concerns between different layers could explain why the feed-forward approximation works well in practice.

on both datasets, the DBM filters much of the pixel information and keeps only the abstract concepts in the leading components of the top representation. On the other hand, the greedy layer-wise training prevents the construction of a solution that makes efficient use of all layers. This effect can be observed in Figure 6 where the DBN learns abstract concepts in the first layer, but is unable to further improve the representation in the second layer.

A remarkable fact is that a single feed-forward pass in the DBM (DBM-FF) is sufficient in order to approach the level of abstraction obtained by letting the DBM converge to its stationary modes. This can be observed in Figure 6 where the DBM-FF almost rejoins the DBM in the second layer both in terms of prediction and reconstruction error. The DBM-FF approach also outperforms the other feed-forward architecture of study (i.e. the DBN) in terms of number of layers necessary to build the abstraction. This result corroborates the capacity of the DBM to organize the solution into a feed-forward hierarchy of increasingly abstract feature extractors.

Our quantitative analysis is further supported by visual inspection of learned filters. Figure 4 shows that the DBM builds a hierarchy of increasingly global filters. On the other hand, the first and the third layers of the DBN contain global filters but the second and the fourth layers map the representation back to something similar to the raw pixel representation. We believe that this alternate mapping between pixel-like features and global features relates to the layer-wise oscillations described before. Also, the higher degree of

independence between top-down and bottom-up contributions (see Figure 7) supports the claim that the DBM better separates concerns across layers.

Admittedly, supervised fine-tuning may mitigate to a certain extent the inconsistencies of the learned feature hierarchy—in that case, the unsupervised pretraining plays merely the role of a regularizer (Erhan et al., 2010)—but the label information is often scarce and typically insufficient to significantly alter the unsupervised representation.

We have observed that the solution learned by the DBN exhibits layer-wise oscillations that prevent the feature hierarchy from quickly extracting the problem relevant subspace from the raw representation. We have observed that the stateful nature of the DBM does not prevent the learning algorithm from organizing the learned solution into a coherent feature hierarchy. From these observations, we may speculate that instead of attempting to build a stateless model by decomposing the problem into feed-forward subcomponents, it is generally better to allocate effort in determining what is the most appropriate neural network structure for a specific problem and letting the feed-forward hierarchy emerge naturally from the undirected model.

References

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*, pages 153–160, 2007.

Mikio L. Braun, Joachim Buhmann, and Klaus-Robert Müller. On relevant dimensions in kernel feature spaces. *Journal of Machine Learning Research*, 9: 1875–1908, Aug 2008.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and David W. Aha. Unsupervised and transfer learning challenge. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 793–800. IEEE, 2011.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

Geoffrey E. Hinton and Terrence J. Sejnowski. Learning and relearning in Boltzmann machines. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1:282–317, 1986.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554, 1982.

Grégoire Montavon, Mikio Braun, and Klaus-Robert Müller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12:2563–2581, 2011.

Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, University of Toronto, 1993.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10 (5):1299–1319, 1998.

Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1064–1071, 2008.

Laurens van der Maaten. A new benchmark dataset for handwritten character recognition. Technical report, Tilburg University, 2009.