Published in Proc. of International Joint Conference on Neural Networks (IJCNN) 2010 in Barcelona, Spain

Topological Features in Locally Connected RBMs

Andreas Müller, Hannes Schulz, and Sven Behnke

***Abstract*— Unsupervised learning algorithms find ways to model latent structure present in the data. These latent structures can then serve as a basis for supervised classification methods. A common choice for unsupervised feature discovery is the Restricted Boltzmann Machine (RBM). Since the RBM is a general purpose learning machine, it is not particularly tailored for image data. Representations found by RBMs are consequently not image-like. Since it is essential to exploit the known topological structure for image analysis, it is desirable not to discard the topology property when learning new representations. Then, the same learning methods can be applied to the latent representation in a hierarchical manner.**

In this work, we propose a modification to the learning rule of locally connected RBMs, which ensures that topological image structure is preserved in the latent representation. To this end, we use a Gaussian kernel to transfer topological properties of the image space to the feature space. The learned model is then used as an initialization for a neural network trained to classify the images. We evaluate our approach on the MNIST and Caltech 101 datasets and demonstrate that we are able to learn topological feature maps.

I. Introduction

Many classifiers rely on well-designed features for high performance. Since hand-crafting such features is infeasible for complex data sets, a variety of methods were developed to extract features from data in an unsupervised manner. As unlabeled data is easy to acquire, methods which explain the variance in the data without known labels are preferred. One mean to this end is modeling the data distribution by introducing latent variables. Generative graphical models, such as Restricted Boltzmann Machines (RBM, [1]) are a popular choice for this purpose. RBMs model correlations of observed variables by introducing binary latent variables (features) which are assumed to be conditionally independent given the observed variables. This restriction is useful because, in contrast to general Boltzmann Machines, a fast learning algorithm exists (Contrastive Divergence [1]). RBMs are generic learning machines and have been applied to many domains, including text, speech, motion data, and images. In the most commonly used form, however, they do not take advantage of the topology of the input space. Especially when applied to image data, fully connected RBMs model long-range dependencies which are known to be weak in natural images [2]. Typically (e.g. [3]), a sparsity constraint is introduced, resulting in local receptive fields. On the negative side, this approach wastes space and time, as large parts of the parameter space have to be "unlearned".

Another way to deal with the sparsity problem is to remove long-range parameters from the model entirely. The advantage of this approach is two-fold: first, local connectivity serves as a prior that matches well to the properties of natural images and, second, the drastically reduced number of parameters makes learning on larger images feasible.

While local receptive fields are well-established in discriminative learning, their counterpart in the generative case, which we call "impact area", is not well understood. In [4], we investigated the capabilities of stacked RBMs and RBM-like graphical models with local impact area (LIRBMs) and lateral connections. We showed that with an equal number of parameters, these models are easier to train and give good classification performance. We further demonstrated that with the help of stacking global consistency can be enforced.

We now apply this architecture to much larger images than common in the RBM literature. To obtain representations of the data that are invariant to small transformations, like translations, rotations, or scalings, we introduce a pooling operation, similar to the common convolutional approach. To do this on higher levels, we have to ensure that the learned representations retain structural properties of natural images. This is not the case in standard RBM approaches. To this end, we propose a modification to the contrastive divergence learning rule that enforces local coherence in hidden representations. This learning rule results in locally similar conditional probability distributions of latent variables. This in turn causes filters with impact areas that are close with respect to the image topology to be similar. Thereby, we introduce a method that constitutes a trade-off between the possibility to learn invariant filters (a property of the widely used convolutional neural network) and the possibility to learn locally adjusted filters (a property of the LIRBM).

We evaluate our approach on the MNIST and Caltech 101 datasets. We are able to learn meaningful features and representations that exhibit local structural properties similar to those in the original images. We further use the learned topological features as an initialization for a locally connected neural network that is trained for classification. We show that this pretraining improves classification performance and shortens training times.

II. Background on Boltzmann Machines (BM)

A BM is an undirected graphical model with binary observed variables $\mathbf{v} \in \{0,1\}^n$ (visible nodes) and latent variables $\mathbf{h} \in \{0,1\}^m$ (hidden nodes). The energy function of a BM is given by

$$E(\mathbf{v}, \mathbf{h}, \theta) = -\mathbf{v}^T W \mathbf{h} - \mathbf{v}^T I \mathbf{v} - \mathbf{h}^T L \mathbf{h} - \mathbf{b}^T \mathbf{v} - \mathbf{a}^T \mathbf{h},$$

where $\theta = (W, I, L, \mathbf{b}, \mathbf{a})$ are the model parameters, namely pairwise visible-hidden, visible-visible and hidden-hidden interaction weights, respectively, and $\mathbf{b}$, $\mathbf{a}$ are the biases

University of Bonn – Computer Science VI, Autonomous Intelligent Systems Group, Römerstraße 164, 53117 Bonn, Germany. Email: {amueller,schulz,behnke}@ais.uni-bonn.de

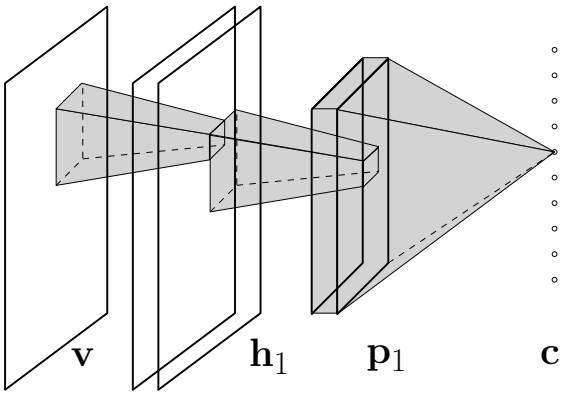

Fig. 1. Overview of our setup. $\mathbf{v}$ is the visible layer of a LIRBM, $\mathbf{h}_1$ is the hidden layer with multiple topographic maps (2 in this example). A sample variable in $\mathbf{h}_1$ is shown to be connected to its respective local impact area in $\mathbf{v}$. $\mathbf{p}_1$ is a pooling layer, invariant to small translations and feature transformations. After training a stack of n LIRBMs one by one, we connect all units in all maps of the last pooling layer $\mathbf{p}_n$ to all class-labelled output units in $\mathbf{c}$ and perform supervised finetuning in the resulting network.

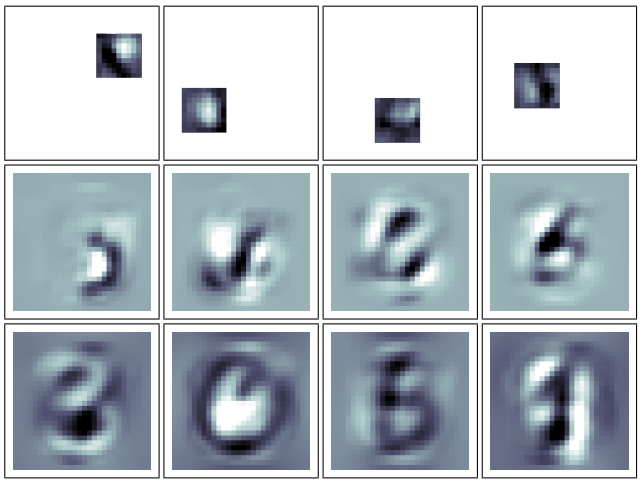

Fig. 2. Selected Features from first, second and third hidden layer of LIRBM. The impact area of units in higher layers increases and captures more complex aspects of the input. The first row shows actual filters, the second and third row show features of single hidden nodes projected down to the visible layer.

of visible and hidden activation potentials. The diagonal elements of I and L are always zero. This yields a probability distribution $p(v)$

$$p(\mathbf{v};\theta) = \frac{1}{Z(\theta)} p^*(\mathbf{v};\theta) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h},\theta)},$$

where $Z(\theta)$ is the normalizing constant (partition function) and $p^*(\cdot)$ denotes unnormalized probability.

In RBMs, I and L are set to zero. Consequently, the conditional distributions $p(\mathbf{v}|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v})$ factorize completely. This makes exact inference of the respective posteriors possible. Their expected values are given by

$$\begin{aligned} \langle \mathbf{v}|h\rangle_p &= \sigma(W\mathbf{h} + \mathbf{b}) \text{ and} \\ \langle \mathbf{h}|v\rangle_p &= \sigma(W\mathbf{v} + \mathbf{b}), \end{aligned} \tag{1}$$

where σ denotes element-wise application of the sigmoid function

$$\sigma(x) = (1 + \exp(-x))^{-1}.$$

In practice, Contrastive Divergence (CD, [1]) is used to approximate the true parameter gradient

$$\partial \ln p(\mathbf{v}) / \partial w_{i,j} = \langle \mathbf{v}^T \mathbf{h} \rangle_+ - \langle \mathbf{v}^T \mathbf{h} \rangle_-$$

by a Markov Chain Monte Carlo algorithm. Here $\langle\cdot\rangle_+$ and $\langle\cdot\rangle_-$ refer to the expected values with respect to the data distribution and model distribution, respectively. The expected value $\langle \mathbf{v}^T \mathbf{h} \rangle_+$ can be calculated in closed form using the formula for $p(h|v)$. This calculation is called the "positive phase". The model distribution $\langle \mathbf{v}^T \mathbf{h} \rangle_-$, on the other hand, can only be approximated using a Markov chain starting from the data and using the transition operators given by Equation (1). This process is termed the "negative phase". Recently, Tieleman [5] proposed a faster alternative, called Persistent Contrastive Divergence (PCD), which employs a persistent Markov chain to approximate $\langle\cdot\rangle_-$. We use PCD throughout this paper.

RBMs can be stacked to build hierarchical models. The training of stacked models proceeds layer-wise by training the high-level models using the activations of the hidden nodes of the already trained layer below as input.

III. Local Impact Restricted Boltzmann Machines

In [4], we introduced a modification to the architectures of Section II called the Local Impact Restricted Boltzmann Machine (LIRBM). LIRBMS have a restricted impact area for each hidden node. To this end, we arranged the hidden nodes in multiple maps, each of which resembles the visible layer in its geometry. As a result, each hidden node h_j can be assigned a position $pos(h_j) \in \mathbb{N}^2$ in the input (pixel) coordinate system. This approach is similar to the common approach in convolutional neural networks [6]. In contrast to the convolutional procedure, we do not require the weights to be equal for all hidden units within one map.

A common problem in training convolutional RBMs is that the over-complete representation of the visible nodes by the hidden nodes allows the filters to learn a trivial identity ([7], [8]). The LIRBM does not suffer from this problem for two reasons: First, we use PCD for training. This means even if a training example can be perfectly reconstructed, there is still a non-zero learning signal. This signal stems from the dependency of the approximated gradient on the state of the persistent Markov chain. Second, due to not sharing weights, for a trivial solution to occur, all filters have to learn the identity separately.

A. Topological Feature Maps

To obtain invariant features, it is common to arrange features in a topological map (e.g. [9], [10], [11]), where similar features are close to each other. All filters are then applied to the input image and responses are grouped together in neighborhoods which are local in the topological map. After

grouping, the features are invariant to transformations which are local in the topology of the feature map.

In our architecture, we now introduce modifications to learn multiple topological maps. Moreover, we endow the feature maps with a topology which matches the input topology. This is in contrast to previous work, where the topology of the features was chosen arbitrarily. Specifically, we arrange the features in the topology that is naturally given by the geometry of each map defined in Section III. The structure of the LIRBM (or RBMs, for that matter), however, is symmetric with respect to hidden units with same impact area. We break this symmetry by indirectly encouraging local filter similarity within maps.

To this end, we propose a modification to the contrastive divergence learning rule. During the positive phase, we simulate local similarity using a local distance kernel. The conditional distribution of the hidden units is given by:

$$p(h_i|v) = \sum_{j \in M(i)} \frac{1}{\sqrt{2\pi}s} e^{-\left(\frac{d(i,j)^2}{2s^2}\right)} \cdot \sigma\left(\sum_k W_{kj} v_k\right). \quad (2)$$

Here, s determines the degree of locality in the topological map, $d(i,j) = \|pos(h_i) - pos(h_j)\|$ denotes the Eucledian distance of two latent variables in the map, and $M(i)$ is the set of all hidden units sharing a map with h_i. The negative phase is left unchanged. The idea behind the local distance kernel is that during the positive phase, we pretend that close latent variables have similar conditional distributions. Contrastive divergence learning fits the model distribution to the data distribution and consequently creates features with locally similar activation patterns. Note that it is not necessary to constrain the features directly: we rather encourage neighboring hidden units within a map to react to the same input patterns. The similarity of features then emerges by virtue of the local statistics of natural images. For inference, since filters are already ordered topologically, the distance kernel is not necessary anymore and the hidden activations are given by Equation (1) again.

B. Higher-Order Topological Feature Maps

Due to the local similarity of features within a map, latent variable activities for a given input have an image-like structure within a map: as in natural images, neighboring positions are strongly correlated. Consequently, a coherent grouping of features that are close within a map is possible. Grouping can, for example, be performed by a simple maximum pooling operation. This operation yields another, more compact representation which is again locally coherent and image-like. As these are the properties of the original input, it is possible to train another LIRBM using the pooled representations as its input. The resulting hierarchy of representations is increasingly invariant to local distortions and translations. Higher-order features can thus represent more complex patterns by combining input from a specific location across all maps in the layer below.

C. Initializing Neural Networks with Topological Features

In many object recognition applications, it was shown that initializing the weight of a neural network with an unsupervised feature extraction algorithm improved the classification results. This procedure is commonly known as unsupervised pretraining and supervised finetuning in the deep learning literature.

We use a locally connected neural network (LCNN, e. g. [12], [13]) for classification. Using the weights learned in a Local Impact Restricted Boltzmann Machine for initialization is straight-forward, since the arrangement of neurons in a LCNN matches the arrangement of hidden nodes in the LIRBM. This makes it possible to directly copy the weights learned by the LIRBM into a LCNN. We then replace the soft-max layers of the LIRBM by simple maximum pooling layers. There are, however, no standard procedures for the back-propagation of error through max-pooling layers when no weight sharing is employed. Here, we use the simplest way of performing gradient descent across the max-pooling layer, by back-propagating the error only to the input node that had the maximum activity in the forward pass. While in principle, the same path could be taken every time the output is calculated, the pre-training largely rules out these pathological cases by providing sensible default filters. Finally, we connect an output layer to the LIRBM by adding random weights between each output (classification) unit and all units in the last pooling layer of the LIRBM, respectively.

We can now proceed with the fine-tuning using gradient descent (backpropagation of error). For this work, we employed batch learning using RPROP [14] with default parameters to minimize the cross-entropy error. First, the output layer, which was not pretrained, is adjusted while all other weights remain fixed. This prevents problems induced by large errors in the randomly initialized weights of the output layer destroying the pre-trained weights in earlier layers. After 50 epochs the output weights have converged and we continue training network as a whole.

D. GPU Acceleration

As the 2D structure of the input maps well to the architecture of Graphics Processing Units (GPUs), we implemented the proposed architecture using the NVIDIA CUDA programming framework. To this end, we created a matrix library which, apart from common dense-matrix operations can operate on sparse matrices in diagonal format (DIA). This enables us to efficiently process large images, where the corresponding dense matrix W would not even fit in memory. Specifically, we implemented $H \leftarrow W^T V$ and $V \leftarrow WH$, where H and V are dense matrices containing hidden/visible activations, respectively, and W is the LIRBM weight matrix. Efficiency is further improved by processing 16 images in parallel. The gradient, $\Delta W \leftarrow VH^T$ is calculated by multiplying all blocks of V and H, where the resulting block is cut by one of the diagonals in ΔW. For maximum pooling operations we made use of routines provided by Alex

LCNN	With Pretraining	Without Pretraining
maps 1,2,4; no pooling	1.4	1.6
maps 1,2,4; max pooling	1.4	1.9
maps 1,4,8; no pooling	1.3	1.3
maps 1,4,8; max pooling	1.2	1.3

TABLE I

MNIST CLASSIFICATION ERROR IN PERCENT

Krizhevsky[1]. Our implementation, which focuses on ease of use by exporting all functionality to Python and seamlessly integrating with NumPy, gives an overall speedup of about 50 when compared to an optimized single CPU version. The code is available at our website[2].

IV. Related Work

In the context of generative models of images, little work has been done to exploit local structure. A well known supervised learning approach that makes use of the local structure of images is the convolutional neural network by LeCun et al. [6]. LeCun's ideas where transfered to the field of generative graphical models by Lee et al. [7] and Norouzi et al. [8]. Their models, which employ weight sharing and max-pooling, discard global image statistics. Our model does not suffer from this restriction. When training landscapes, for example, our model would be able to learn, even on the lowest layer, that there is always sky depicted in the upper half of the image. In the neural networks context, architectures which exhibit a similar local structure to ours were investigated by Behnke [12] and have been applied to natural images on a large scale by Uetz [15]. The latter architecture proved to be competitive to state-of-the-art in object recognition.

In our, as well as in convolutional architectures, the activities of each feature map are a filtered version of the input. However, in convolutional architectures, the same filter is applied at each position of an image. Our topological features, on the contrary, are adjusted specifically to the local statistics at a given position.

Our model also shares some of the ideas with Kavukcuoglu [9]: The authors learned a set of filters, placed on a two dimensional grid, using a sparse coding algorithm. Features are first learned on a topographical map and then grouped according to proximity in the topology. While the topology in [9] is artificially constructed, with no relation to the input, our topology reflects and is adjusted to the topology of the input space.

Other methods for topological filters are based on non-negative matrix factorization (NMF, [16], [11]). However, both methods do not learn features specific to the spatial layout of the input.

Pretraining deep neural networks with unsupervised methods is covered extensively in the literature. Good surveys of the state-of-the-art can be found in the papers of Bengio [17] and Erhan et. al. [18].

[1]`http://www.cs.utoronto.ca/~kriz`

[2]`http://www.ais.uni-bonn.de/deep_learning`

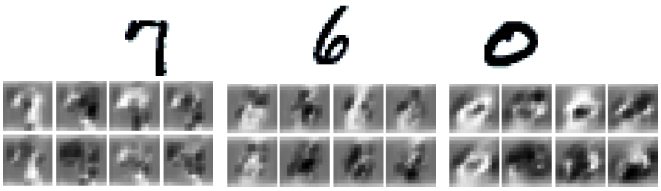

Fig. 5. MNIST samples and hidden activations. The first row shows sample MNIST digits. The second row displays the respective hidden representations. Each digit is represented with eight maps.

V. Results

In our implementation, we used maximum pooling in 2×2 non-overlapping windows. The width s of the distance kernel was set such that only direct neighbors influenced each other.

We trained our model on the MNIST dataset of handwritten digits [6] and on the Caltech 101 [19] object categories database. Training on both datasets was purely unsupervised.

A. MNIST

1) Unsupervised Feature Extraction: The MNIST dataset consists of 60,000 training and 10,000 test images. The images are centered gray scale images of handwritten digits. The size of each image is 28×28 pixels. We trained our hierarchical model with three layers of LIRBMs, where the layers had four, eight and 16 maps of topological features, respectively. The size of the local impact area was set to 9×9 pixels on all layers. Training the model took approximately 30 minutes on GPU. We first analyzed the learned feature maps. As expected, the features within a map are locally similar and vary smoothly between locations. In regions where the data exhibits a large variance, each map concentrates on a different aspect of the data. We show two learned feature maps of the lowest layer in Figure 3. Higher-layer features exhibit similar continuity properties.

Apart from visualizing the weight matrices directly, we can also project down activation patterns corresponding to a single high-level unit turned on, and the others clamped to zero. To calculate the hidden activations of the lower layer, we have to reverse the max-pooling operation. This is done by supersampling and dividing the resulting activations by the size of the pooling window. After projecting down all filters of a given layer, we subtracted the mean from the resulting activities in the image domain for visualization purposes. We show sample features in Figure 2. Features of the first hidden layer represent simple edge filters, while the filters of the second hidden layer represent parts of numbers. The third-layer features have a impact area large enough to cover the whole image. Note, that despite the ad-hoc reversal of max-pooling, filters are still quite pronounced.

2) Classification: In order to assess the fitness of the extracted features for classification, we train a locally connected neural network, as described in Section III-C. Since we aim at comparing different learning methods, we do not augment the training set with deformed or translated versions of the training data, which generally guarantees performance improvements (e. g. [6]).

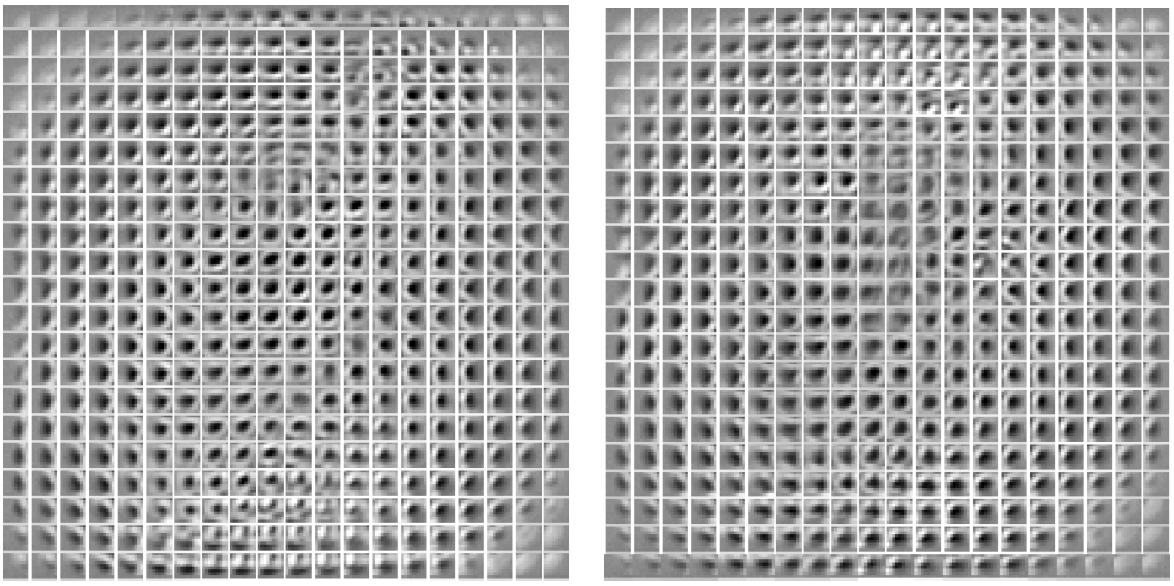

Fig. 3. Two topographic feature maps learned on the MNIST dataset in our LIRBM model. The filters are clearly specific to the properties of the dataset in their respective areas. Features vary smoothly with the topology but differ between maps.

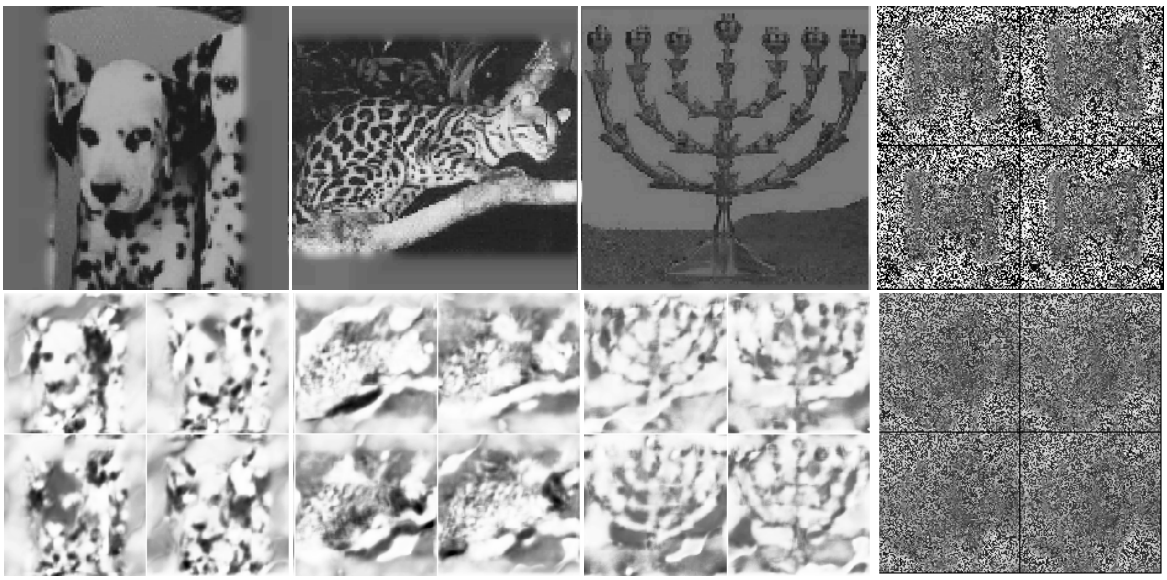

Fig. 4. Topological learning on Caltech 101. The three leftmost columns display Caltech images (top) and their first-layer representations (bottom). The four variations are the result of the four different filter maps. The rightmost column shows first-layer activations of the same LIRBM trained without topographic feature maps.

We compare our approach with a vanilla fully connected neural network, a locally connected network without pretraining and a pretrained network where the pretraining did not use topological filters.

Using a parameter scan we found that the best result for using a vanilla neural network are achieved with one hidden layer containing 1000 neurons. Using this setup, we obtained a test error of 1.7%, which serves as a baseline.

We used locally connected networks with two hidden layers and two, four, eight or 16 maps per layer. The size of the local receptive fields (or impact areas in the LIRBM) was kept fixed to 9×9. If pooling was applied, the size of the pooling window was set to 2×2. Table I summarizes the obtained results.

From the table it is clear that locally connected networks have an advantage over fully connected networks. The use of max-pooling layers does not significantly influence the classification rate. Using maximum pooling, however, makes the model more scalable since the upper layers decrease in size and less parameters have to be learned and stored. We find that the pre-trained neural networks perform best. The influence of topological filters was also analysed. We found

that networks pretrained without topological filters performed worse (1.32%) than networks which used topological filters (1.18%). This tendency is expected to strengthen for deeper networks as more dissimilar filters are pooled.

B. Caltech 101

Due to the local impact areas of our model, it is feasible to train even on large images, which is not possible using fully connected RBMs. We trained two models on the Caltech 101 dataset. We preprocessed the images by converting them to gray scale and fit them into a 128×128 pixel image, keeping the aspect ration constant. The images were padded to square aspect ratio using their respective average gray level. We further faded the image border into the padding to remove side effects caused by the image edges.

To train on natural images, it is necessary to use Gaussian visible units (see [20]). For this purpose, we normalized the images such that each pixel has zero mean and unit variance. The conditional distribution of the visible units after this normalization is given by

$$p(\mathbf{v}|\mathbf{h}) \propto \mathcal{N}(W\mathbf{h}, 1).$$

We trained one standard LIRBM model and one using the modified learning rule for topological feature maps. Both models consisted of three hidden layers with four, eight and eight maps, respectively. The training time for this model was about one hour. Figure 4 shows sample inputs and activations from both models. Only the hidden activations of the model trained with topographic filter maps retain the image structure and partly preserve local texture. Again, different filter maps concentrate on different aspects of the image.

VI. Conclusion

In this work, we addressed the problem of learning locally coherent representations in locally connected Restricted Boltzmann Machines. Locality is a desirable property of learning algorithms especially on images, since local algorithms are faster and can retain basic structural properties of the data. However, when training a locally connected RBM in a naïve way, the resulting representations do not reflect local correlations present in natural images. This loss of structure prevents stacking of such models.

Here, we present a novel modification of the contrastive divergence learning rule. We map the topology of the input space to the hidden units and thereby encourage the learning algorithm to learn similar features at neighboring positions. The resulting activations then exhibit the desired properties.

We evaluated our method on the MNIST and Caltech 101 datasets and found that our topological features are useful for classification and generate significantly more image-like hidden representations when compared to models trained without topological feature maps.

In future work, we are going to extend our hierarchical learning architecture to (semi-) supervised fine-tuning using deep Boltzmann machines and third order methods.

References

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

[2] J Huang and D Mumford. Statistics of natural images and models. In *CVPR*, 1999.

[3] MA Ranzato, L Boureau, and LeCun Y. Sparse feature learning for deep belief networks. *Advances in neural information processing systems*, 20, 2007.

[4] H Schulz, A Müller, and S Behnke. Exploiting local structure in stacked Boltzmann machines. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2010.

[5] T Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *ICML*, 2008.

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

[7] H Lee, R Grosse, R Ranganath, and Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

[8] M Norouzi, M Ranjbar, and G Mori. Stacks of Convolutional Restricted Boltzmann Machines for Shift-Invariant Feature Learning. In *CVPR*, 2009.

[9] K Kavukcuoglu, MA Ranzato, R Fergus, and Y Le-Cun. Learning invariant features through topographic filter maps. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009*, pages 1605–1612, 2009.

[10] Y Karklin and MS Lewicki. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*, 457(7225):83–86, 2008.

[11] K Hosoda, M Watanabe, H Wersing, E Körner, H Tsujino, H Tamura, and I Fujita. A model for learning topographically organized parts-based representations of objects in visual cortex: Topographic nonnegative matrix factorization. *Neural Computation*, 21(9):2605–2633, 2009.

[12] S Behnke. *Hierarchical neural networks for image interpretation*. Springer-Verlag, 2003.

[13] R Uetz and S Behnke. Large-scale Object Recognition with CUDA-accelerated Hierarchical Neural Networks. In *Proc. Intelligent Computing and Intelligent Systems*, 2009.

[14] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. pages 586–591, 1993.

[15] R Uetz and S Behnke. Locally-Connected Hierarchical Neural Networks for GPU-accelerated Object Recognition. In *NIPS 2009 Workshop on Large-Scale Machine Learning: Parallelism and Massive Datasets*, 2009.

[16] T Zhang, B Fang, Y Tang, G He, and J Wen. Topology preserving non-negative matrix factorization for face recognition. *IEEE Transactions on Image Processing*, 17(4):574, 2008.

[17] Y Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[18] D Erhan, P Manzagol, Y Bengio, S Bengio, and P Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Proc. Artificial Intelligence and Statistics (AISTATS'09)*, pages 153–160, 2009.

[19] L Fei-Fei, R Fergus, and P Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[20] G Hinton and R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504, 2006.