Texture Modeling with Convolutional Spike-and-Slab RBMs and Deep Extensions

Heng Luo **Pierre Luc Carrier** **Aaron Courville** **Yoshua Bengio**
{heng.luo, pierre-luc.carrier, aaron.courville, yoshua.bengio}@umontreal.ca
Department of Computer Science and Operations Research
University of Montreal
Montreal, H3C 3J7

Abstract

We apply the spike-and-slab Restricted Boltzmann Machine (ssRBM) to texture modeling. The ssRBM with tiled-convolution weight sharing (TssRBM) achieves or surpasses the state-of-the-art on texture synthesis and inpainting by parametric models. We also develop a novel RBM model with a spike-and-slab visible layer and binary variables in the hidden layer. This model is designed to be stacked on top of the ssRBM. We show the resulting deep belief network (DBN) is a powerful generative model that improves on single-layer models and is capable of modeling not only single high-resolution and challenging textures but also multiple textures with fixed-size filters in the bottom layer.

1 Introduction

Texture processing is an essential components of scene understanding in human vision. Natural images can be seen as a large mixture of heterogeneous textures. Thus, to some extent, progress in modeling natural images requires that we make progress in modeling textures. To this end, texture modeling has been an active research area of machine learning, computer vision and graphics during the past five decades. Although non-parametric approaches (Lin *et al.*, 2006) have made significant progress in synthesizing textures from example images, capturing the statistical properties of textures via a probabilistic model remains an active area of inquiry. Such probabilistic models are important for modeling natural images (Heess *et al.*, 2009) and understanding human vision (Zhu *et al.*, 2000).

In this work we consider a probabilistic model of textures based on the spike-and-slab Restricted Boltzmann Machine (ssRBM) (Courville *et al.*, 2011a,b). The ssRBM has previously demonstrated the ability to generate samples of small natural images that preserved much of their statistical structure (Courville *et al.*, 2011b), suggesting that the ssRBM may be well suited to texture modeling. Following the recent exploration of Boltzmann machines for textures (Kivinen and Williams, 2012), we have trained ssRBMs with tiled-convolution weight sharing (Gregor and LeCun, 2010; Le *et al.*, 2010) on the Brodatz-texture images[1]. Tiled convolution allows weight sharing in filters with non-overlapping receptive fields.

Kivinen and Williams (2012) concentrated their quantitative evaluation of the texture models on a subset of the Brodatz textures that exhibit strong spatial invariance, i.e. textures largely consisting of a regular repeating pattern. While this is an important problem in its own right, most natural textures (i.e. those associated with a natural-looking world) exhibit significant spatial non-stationarity and features with a wide spatial frequency range. One popular way to deal with images with a wide spatial frequency range is to decompose the frequencies using, for example, a Laplacian image pyramid. However, spatial pyramid-based methods tend to relegate the modeling long-range spatial dependencies to the low-spatial-frequency pipeline, and would likely have trouble capturing features that interact across spatial resolutions (eg. long thin edges). We propose that convolutional *deep* belief networks based on ssRBM are well suited to model these natural textures. In particular, by increasing the effective receptive field with depth, we can use higher layers of the model to efficiently communicate information such as phase to spatially isolated parts of the first layer model.

Appearing in Proceedings of the 16^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2013, Scottsdale, AZ, USA. Volume 31 of JMLR: W&CP 31.

[1] http://www.ux.uis.no/~tranden/brodatz.html

Deep belief networks also have another important property that we find useful in the context of texture modeling. As argued by Hinton (2012), Le Roux and Bengio (2008) and Tijmen Tieleman (2012, personal communication), training the lower layers by contrastive divergence (CD) (Hinton *et al.*, 2006) allows the lower layers to concentrate on modeling local features of the data. We have found best results by training the lower layers by CD and the uppermost layer by a closer approximation to maximum likelihood, such as persistent contrastive divergence (PCD) (Younes, 1999; Tieleman, 2008), thus promoting a better division of labour between the layers of the DBN. On this account, the ssRBM offers an important advantage over other similar models in the literature. Unlike models such as the mcRBM (Ranzato and Hinton, 2010) and mPoT (Ranzato *et al.*, 2010a), the structure of the ssRBM makes it readily amenable to CD training through Gibbs sampling.

Our contributions are, first, the exploration of a tiled-convolutionally trained ssRBM (TssRBM) texture model and its objective comparison with the other similar models in the literature. We show that the TssRBM is competitive with the state-of-the-art on texture synthesis and inpainting tasks on a selection of Brodatz textures. Second, we develop a novel RBM model with a spike-and-slab visible layer and binary variables in the hidden layer. This model is designed to be stacked on top of the TssRBM within a deep belief network configuration, with each layer trained convolutionally with a greedy layer-wise pretraining strategy. We demonstrate how the resulting two and three-layer DBN (the third layer is a standard RBM) models are able to encode longer term dependencies in the higher layers while simultaneously recovering more detailed structure in the CD-trained lower layer, all of which translates to superior texture model performance – particularly when the textures being modeled exhibit strong non-stationarity. Finally, we show how depth helps in learning a generative model of multiple textures by using fixed-size filters in the bottom layer. Kivinen and Williams (2012) introduced a model of multiple textures, however they made use of label information in the training process alleviating the difficult learning problem of constructing multiple modes to represent each texture. In this work, we show how a deep belief network based on the ssRBM is capable of learning to model multiple textures based on purely unsupervised training.

2 Previous Work

The problem of texture synthesis has been extensively studied in the computer vision community for decades (Zhu *et al.*, 2000). Probably the most popular texture synthesis strategies are currently example-based or nonparametric methods (Wei *et al.*, 2009). These typically compose a target image with transformed versions of patches drawn from the target texture. While these methods are flexible, they are unlikely to be readily applicable to natural textures, where some aspects of the statistical structure are global in scope.

The Gaussian RBM (Welling *et al.*, 2005) models real-valued observations by adding quadratic terms on the visible units to the RBM energy function. One limitation of the Gaussian RBM is that changing its hidden unit activations only changes the conditional mean of the visible units. For modeling natural images, there is evidence that it is important to allow the hidden unit configuration to capture changes in the covariance between pixels. This observation has motivated several of the models discussed below as well as the ssRBM. The product of Student's T-distributions (PoT) model (Welling *et al.*, 2003) is an energy-based model where the conditional distribution over the visible units, conditioned on the hidden variables, is a multivariate Gaussian (non-diagonal covariance) and the complementary conditional distribution over the hidden variables, given the visibles, is a set of independent Gamma distributions. The PoT model has recently been generalized to the mPoT model (Ranzato *et al.*, 2010b) to include nonzero Gaussian means by the addition of Gaussian RBM-like hidden units. Kivinen and Williams (2012) explored the "Multi-Texture Boltzmann Machine" (Multi-Tm), training a single large Gaussian RBM (up to 256 feature maps per tiling position, as opposed to 32 maps per tiling position) on multiple textures. In modeling multiple textures, Kivinen and Williams (2012) used label information during the training process to enable the model to focus on a single texture class at a time. In section 5.3, we show how we can use a deep belief network, based on the ssRBM, to learn a model of multiple textures with fixed-size filters in the bottom layer and using no label information at all.

In addition to validating the ssRBM as the basis of a texture model, we also set out to study the impact of adding layers to the tiled-convolutional ssRBM model, in order to see if depth can help maintain coherence of large scale texture features. Recent work (Ranzato *et al.*, 2011) has shown that stacking additional RBM layers on top of an mPoT model (also trained using tiled-convolutional weight sharing) can have a dramatic impact on the ability of the model to generate globally coherent natural image samples. Findings such as these motivated our attempt to use depth to synthesize textures with increased global coherence.

3 Spike-and-Slab RBM

The ssRBM describes the interaction between three sets of random variables: the real-valued visible random vector $v \in \mathbb{R}^D$ representing the observed data of dimension D, the set of binary "spike" random variables $h \in \{0,1\}^N$ and the real-valued "slab" random variables $s \in \mathbb{R}^N$. The ssRBM has the interpretation that, with N hidden units, the ith hidden unit is associated with *both* an element h_i of the binary vector and an element s_i of the real-valued variable. In this work we will concern ourselves with the ssRBM formulation referred to as the μ-ssRBM (Courville *et al.*, 2011b) with the associated energy function:

$$E_1(v,s,h) = -\sum_{i=1}^{N} v^T W_i s_i h_i \; + \frac{1}{2} v^T \left(\Lambda + \sum_{i=1}^{N} \Phi_i h_i\right) v + \frac{1}{2}\sum_{i=1}^{N} \alpha_i s_i^2 \; - \sum_{i=1}^{N} \alpha_i \mu_i s_i h_i \; - \sum_{i=1}^{N} b_i h_i + \frac{1}{2}\sum_{i=1}^{N} \alpha_i \mu_i^2 h_i,$$

where $W_i \in \mathbb{R}^D$ denotes the ith weight (or feature) vector, b_i is a scalar bias associated with the spike variable h_i, μ_i and α_i are respectively a mean and precision parameter associated with the random slab variable s_i, Λ is a diagonal precision matrix on the visibles v, and Φ_i is an digonal matrix as an h_i-gated contribution to the precision on v. As is standard with energy-based models, the joint probability distribution over v, $s = [s_1, \ldots, s_N]$ and $h = [h_1, \ldots, h_N]$ is specified as: $p_1(v,s,h) = \frac{1}{Z}\exp\{-E_1(v,s,h)\}$, where Z is the normalizing partition function. Note, we use the subscript notation E_1 to denote this as the 1st-layer energy function.

An interesting property of the ssRBM is that despite having higher-order interactions of variables, the model maintains the bipartite graph structure of the standard restricted Boltzmann machine where the ith hidden unit consists of the product of the random variables s_i and h_i. This property makes it easy to use efficient block Gibbs sampling, using the conditionals:

$$\begin{aligned}
p_1(v \mid s,h) &= \mathcal{N}\left(C_{v|s,h}\sum_{i=1}^{N} W_i s_i h_i \;,\; C_{v|s,h}\right), \\
P_1(h \mid v) &= \prod_{i=1}^{N} \sigma\left(\frac{1}{2}\alpha_i^{-1}(v^T W_i)^2 + v^T W_i \mu_i - \frac{1}{2} v^T \Phi_i v + b_i\right), \\
p_1(s \mid v,h) &= \prod_{i=1}^{N} \mathcal{N}\left(\left(\alpha_i^{-1} v^T W_i + \mu_i\right) h_i \;,\; \alpha_i^{-1}\right),
\end{aligned}$$

where $\mathcal{N}(\mu, C)$ denotes a Gaussian distribution with mean μ and covariance C, σ represents a logistic sigmoid, and $C_{v|s,h} = \left(\Lambda + \sum_{i=1}^{N} \Phi_i h_i\right)^{-1}$ is the diagonal conditional covariance matrix.

ssRBM training: Like the standard RBM, learning and inference in the ssRBM is rooted in the ability to efficiently draw samples from the model via block Gibbs sampling. In training the ssRBM we are free to use either CD, PCD and fast persistent contrastive divergence (FPCD) (Tieleman, 2008). CD training involves approximating the negative phase component of the likelihood gradient by a few steps (often just one) of Gibbs sampling away from the data presented in the positive phase. In PCD, one maintains a persistent Markov chain to approximate the negative phase and simulates a few Gibbs steps between each parameter update. These samples are then used to approximate the expectations over the model distribution $p_1(v,s,h)$. Details regarding PCD training of the ssRBM are available in Courville *et al.* (2011a).

Our use of block Gibbs sampling marks an important distinction between our approach to learning and that used by Kivinen and Williams (2012), who use Hybrid Monte Carlo (HMC) (Neal, 1994) to draw samples from the model distribution. Their use of HMC is likely motivated by their need to train models such as the PoT model where the conditional over visible vectors do not factorize and hence is not amenable to efficient block Gibbs sampling. Our use of Gibbs sampling from the ssRBM model also makes it amenable to CD training. As we show in the experiments with deeper models, the use of CD training is crucial to achieving our best results.

Tiled-Convolutional ssRBM: Gregor and LeCun (2010) introduced tiled-convolutional weight sharing (Ranzato *et al.*, 2010a; Le *et al.*, 2010) and is similar to convolutional weight sharing (LeCun *et al.*, 1998; Desjardins and Bengio, 2008; Lee *et al.*, 2009) except that spatially neighboring features (with overlapping receptive fields) do not share weights. Within the tiled-convolutional structure, every specific filter tiles the input images without overlap. The tiled-convolution architecture has the advantage that it results in less correlation in the activations of the hidden units in comparison to traditional convolutional achitectures which could lead problems when training.

To make comparisons easier, our TssRBM uses the same architecture as Kivinen and Williams (2012) including the same receptive field size of 11×11 and the same diagonal tiling pattern with a stride of one pixel (neighboring receptive fields are offset by one pixel). This diagonal tiling (which reduces considerably the number of free parameters) makes for 11 sets of filters (one for each offset). We also kept constant the number of filters (32 per set).

4 Spike-and-Slab Deep Belief Network

In this section, we describe how we extend the TssRBM to a hierarchical generative model in the form of a deep belief network (DBN). In the satandard pretraining procedure of DBN(Bengio *et al.*, 2007), training the bottom layer ssRBM, either by CD, PCD or FPCD is straightforward and discussed above in Sec. 3. We now consider the form of the model we intend to stack on top of the ssRBM.

Following the DBN approach, we express the ssRBM model as

$$p_1(v) = \sum_{s,h} p_1(v|s,h)p_1(s,h).$$

As discussed in the previous section, due to the factorial nature of $p_1(v|s,h)$, it is convenient to consider this the bottom layer of our DBN and focus on how to model the spike-and-slab latent state. Let $\hat{p}(v)$ denote the data distribution. We introduce another higher-layer model of the spike-and-slab state $p_2(s,h)$ to model the aggregated posterior distribution, $\hat{p}(s,h)$, of the ssRBM

$$\hat{p}(s,h) = \sum_{v} p_1(s,h|v)\hat{p}(v)$$

If $p_2(s,h)$ models the aggregated posterior $\hat{p}(s,h)$ better than does $p_1(s,h)$ (defined by the ssRBM), then adding the second layer can improve the model of the training data (Hinton *et al.*, 2006).

Formally, the two layer DBN is,

$$p_{\text{DBN}}(v) = \sum_{s,h} p_1(v|s,h)p_2(s,h)$$

From a generative perspective, the sampling procedure consists of generating a sampling pair (s,h) from the top (second here) layer, followed by mapping them to image space through $p_1(v|s,h)$.

In specifying the form of $p_2(s,h)$, we follow the common practice of using another model of the RBM family to model the distribution over s and h. To this end, we introduce a 2nd-layer latent variable model: $p_2(s,h,g)$, which models the aggregate posterior $\hat{p}(s,h)$ through a hidden a binary random vector: $g \in \{0,1\}^M$. We choose to use a binary hidden layer in order to transition to a more standard binary representation. When we include a third layer to the DBN, then that layer will be formed by training a standard binary-binary RBM.

The energy function of the second layer model is defined as follows:

$$E_2(s,h,g) = -\sum_{i=1}^{N}\sum_{j=1}^{M} g_j U_{ij} s_i h_i - \sum_{j=1}^{M} \rho_j g_j + \frac{1}{2}\sum_{i=1}^{N} \alpha_i s_i^2 - \sum_{i=1}^{N} \alpha_i \mu_i s_i h_i + \frac{1}{2}\sum_{i=1}^{N} \alpha_i \mu_i^2 h_i - \sum_{i=1}^{N} b_i h_i$$

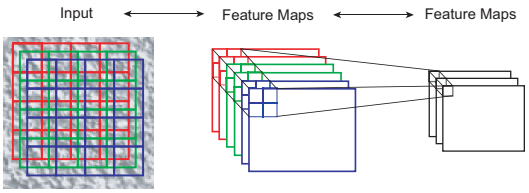

Figure 1: The architecture of the lowest two layer. The first layer possesses tiled-convolutional weight sharing in a diagonal arrangement (tilings are represented by different colors). Each second layer unit has a 2×2 receptive field over all the feature maps in the first hidden layer. The second layer is arranged with traditional convolutional weight sharing and a stride of 1.

where U_{ij} refers to the ijth element of the weight matrix encoding the interactions between g_j and spike-and-slab variables h_i and s_i respectively. The term ρ_j controls the bias on the binary g_j. All other parameters have the same interpretation as their first layer analogues.

Similar to the standard ssRBM, the conditionals $P_2(h \mid g)$, $p_2(s \mid h,g)$ and $P_2(g \mid s,h)$ are factorial and given by:

$$P_2(h_j \mid g) = \sigma\left(\frac{1}{2}\alpha_i\left(\alpha_i^{-1}\sum_{j=1}^{D} g_j U_{ij} + \mu_i\right)^2 + b_i - \frac{1}{2}\alpha_i\mu_i^2\right)$$

$$p_2(s \mid g,h) = \mathcal{N}\left(\left(\alpha_i^{-1}\sum_{j=1}^{M} g_j U_{ij} + \mu_i\right)h_i \;,\; \alpha_i^{-1}\right)$$

$$P_2(g_j \mid s,h) = \sigma\left(\sum_{i=1}^{N} U_{ij} s_i h_i + \rho_j\right)$$

We refer to this model, when stacked on top of a TssRBM (a ssRBM with tiled-convolution weight sharing), as the TssDBN.

2nd-layer training: After pretraining the first layer ssRBM, given training data v, we sample $\tilde{h}$ from $P_1(h \mid v)$, then take $\mathbb{E}_{p_1(s|v,\tilde{h})}[s]$ and $\mathbb{E}_{p_1(h|v)}[h]$ as the new training data to train the second layer. Just as we do for the bottom-layer ssRBM, we train this second-layer RBM with either PCD or with CD. We typically see best results if we train with PCD for the top-layer model and with CD for all other layers.

Sampling and inference in 2-layer TssDBN: Once the second layer has been trained with PCD it can be used to generate samples. We run Gibbs sampling in the top layer, getting the sample $\tilde{g}$. Next, we sample $\tilde{h}$ from $P_2(h \mid \tilde{g})$ then pass $\mathbb{E}_{p_2(s|\tilde{g},\tilde{h})}[s]$ to the bottom layer. The final sample in image sapce will be generated through $p_1(v \mid \tilde{h}, \mathbb{E}_{p_2(s|\tilde{g},\tilde{h})}[s])$ without adding noise. Inference in TssDBN is exactly the same to the process of converting training data into the new representation. Given v, we sample $\tilde{h}$ from $P_1(h \mid v)$,

then pass $\mathbb{E}_{p_1(s|v,\tilde{h})}[s]$ and $\mathbb{E}_{P_1(h|v)}[h]$ to higher layer.

Convolutional Structure: The second layer possesses a convolutional weight sharing structure (not tiled-convolutional). Based on our use of patches of size 98×98 randomly cropped from the texture images, the tiling structure of the first layer model results in a set of 32×11 feature maps of size 8×8 (the receptive field size was 11×11). Second layer hidden units are each connected to all 32×11 feature maps with the same 2×2 receptive field across all feature maps. Using a stride length of 1, this implies that each second layer feature is associated with a feature map of size 7×7. For our experiments with a 3-layer model, we keep the same convolutional weight sharing structure for the third layer and use receptive fields of size 2×2.

5 Experiments

We evaluate our texture models on 8 texture images (D4, D6, D16, D21, D53, D68, D77 and D103) from the Brodatz texture dataset. Acording to Lin *et al.* (2006), we can roughly classify them into 4 different types, regular textures (D6, D21, D53, D77), near-regular textures (D16, D103), irregular textures (D68) and stochastic textures (D4). The regular textures are simpler: shallow models (such as mPoT, Gaussian RBM and ssRBM with tiled-convolutional weight sharing) are able to model them with high fidelity. However, the other textures (D4, D16, D68 and D103) remain challenging for shallow models. We show that 2-layer TssDBNs give better results.

5.1 TssRBM texture modeling

In this section, we compare TssRBM and 2-layer TssDBN with other related models in the literature. We base our comparison on the results reported in Kivinen and Williams (2012). To provide a fair comparison, we follow the general experimental protocol established by Heess *et al.* (2009) and Kivinen and Williams (2012). Specifically, we rescaled the original 640×640 textures (all but D16) to either 480×480 (D4, D21 and D77) or 320×320 (D6, D53, D68 and D103). Each texture image was divided into a top half for training and a bottom half used for testing. For additional comparison, we implemented a 2-layer DBN (2-layer TGDBN) with a tiled-convolutional Gaussian RBM as the bottom layer. For sampling from this 2-layer TGDBN, we run Gibbs sampling in the top layer and thus project the samples into the image samples without adding noise in the bottom layer. All of these models are implemented with Theano (Bergstra *et al.*, 2010).

We report the performances of the TssRBM, 2-layer TGDBN and 2-layer TssDBN on two tasks: texture synthesis and inpainting. All models, in all experiments, are trained on 98×98 sized patches randomly cropped from the preprocessed training texture images which are normalized to have zero mean and standard deviation of 1. We use a minibatch size of 64.

The TssRBM is trained with FPCD. For deep models, we always pretrain the lower layer with one step CD and train the top layer with PCD (We find that in the higher layer RBMs, the mixing of the negative phase Gibbs chain is relatively fast, so we use PCD). In both PCD and FPCD training processes, at the beginning of learning the persistent chains are initialized with noise and for some textures (especially for those regular textures) restarting the Markov chains with a small probability, like 0.01, seems advantageous to further promote mixing. After training, we aply our models to the following two task: texture synthesis and inpainting.

Texture Synthesis: For this task we generate unconstrained samples from our models by the usual DBN generative procedure[2]. Following Kivinen and Williams (2012), after a large number of "burn-in" samples, we collected 128 samples of size 120×120 for both the 1-layer and 2-layer models. A quantitative measure of the quality of the samples is provided by the Texture Similarity Score (TSS) (Heess *et al.*, 2009), comparing each generated sample with the test patches from the test region of the image. For a sample s and test texture x, the TSS is given by the maximum normalized cross correlation (NCC):

$$TSS(s, x) = \max\left\{ \frac{x_1^T s}{\|x_1\|\|s\|}, \ldots, \frac{x_L^T s}{\|x_L\|\|s\|} \right\},$$

where x_i denotes patch i within the test region of the image and L is the number of possible unique patches in the test region. A patch (and sample) of size 19×19 was used to compute the score. We only use TSS for those regular textures (D6, D21, D53, D77). Fig. 2 compares images of textures synthesized by some of the methods under consideration. Table 1 provides a quantitative comparison based on the TSS and shows that the TssRBM-based models are competitive with these other probabilistic models of texture.

Inpainting: The inpainting (constrained texture synthesis) task requires the models to generate a texture which is consistent with a given boundary. Following Kivinen and Williams (2012), we randomly cut 76×76 texture patches from the test texture images and set the center (54×54) to zeros. The resulting images as the inpainting frames were fed to our models. The inpainting was done by running 500 Gibbs

[2]Gibbs sampling in the top-level RBM, followed (in the case of deep models) by stochastic projection (except for the visible units, as usual, and except for the slab units, where we take the expectation) in image space.

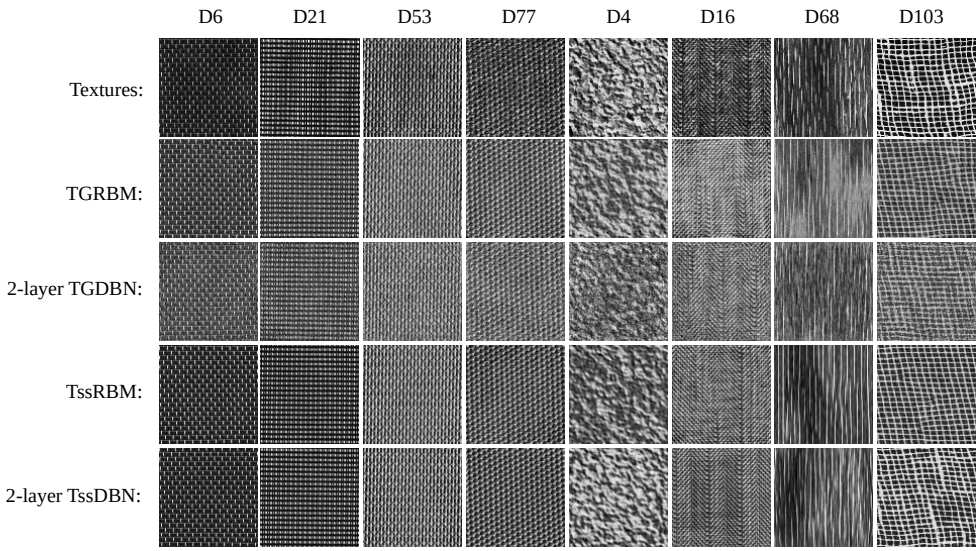

Figure 2: Examples of texture synthesis for the models under consideration (rows) for different textures (columns). The top row has original data.

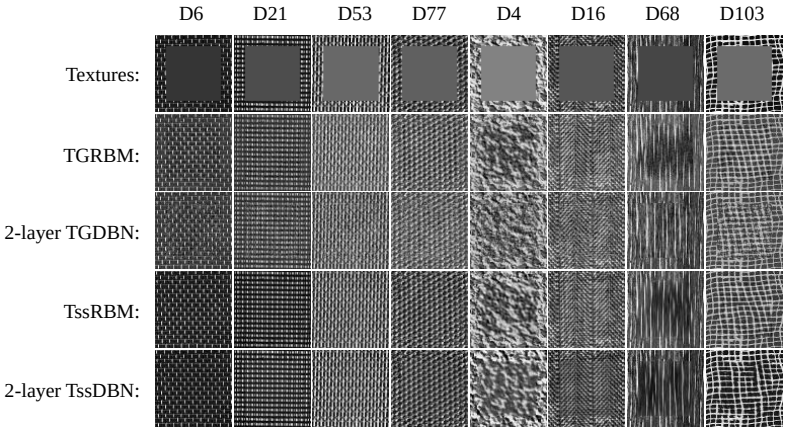

Figure 3: Examples of texture inpainting for the models under consideration (rows) for different textures (columns).

Synthesis	D6	D21	D53	D77
Bi-FoE	0.7573 ± 0.0594	0.8710 ± 0.0317	0.8266 ± 0.0869	0.6464 ± 0.0215
TmPoT	0.9329 ± 0.0356	0.8961 ± 0.0696	0.8527 ± 0.0559	**0.8699 ± 0.0080**
TPoT	0.5641 ± 0.0916	0.7388 ± 0.1055	0.7583 ± 0.1082	0.6870 ± 0.0973
T-GaussRBM	0.9301 ± 0.0207	0.8901 ± 0.0792	0.8485 ± 0.0606	**0.8663 ± 0.0084**
2-layer TGDBN	0.8689 ± 0.0187	0.9108 ± 0.0135	0.9030 ± 0.0111	0.7799 ± 0.0177
Multi-Tm (256)	0.9304 ± 0.0280	0.9346 ± 0.0205	0.9231 ± 0.0103	0.8610 ± 0.0096
TssRBM	0.9365 ± 0.0468	**0.9482 ± 0.0249**	**0.9412 ± 0.0215**	0.8410 ± 0.0121
2-layer TssDBN	**0.9516 ± 0.0164**	**0.9465 ± 0.0322**	**0.9499 ± 0.0264**	**0.8638 ± 0.0161**

Inpainting	D6	D21	D53	D77
Efros&Leung	0.8524 ± 0.0318	0.8566 ± 0.0344	0.8558 ± 0.0578	0.6012 ± 0.0760
TmPoT	0.8629 ± 0.0180	0.8741 ± 0.0116	0.8602 ± 0.0234	**0.7668 ± 0.0322**
TPoT	0.8446 ± 0.0172	0.8609 ± 0.0275	0.8935 ± 0.0159	0.6379 ± 0.0373
T-GaussRBM	0.8578 ± 0.0160	0.8662 ± 0.0185	0.8494 ± 0.0233	**0.7642 ± 0.0267**
2-layer TGDBN	0.8096 ± 0.0203	0.8485 ± 0.0198	0.8470 ± 0.0246	0.6696 ± 0.0405
Multi-Tm (256)	0.8452 ± 0.0173	0.8673 ± 0.0103	0.8554 ± 0.0284	0.7328 ± 0.0615
TssRBM	**0.8881 ± 0.0227**	**0.9119 ± 0.0139**	**0.9156 ± 0.0237**	**0.7627 ± 0.0314**
2-layer TssDBN	**0.8894 ± 0.0246**	**0.9060 ± 0.0160**	**0.9242 ± 0.0285**	**0.7738 ± 0.0232**

Table 1: A comparison of TssRBM and 2-layer TssDBN results with other models. All reported results other than 2-layer TGRBM and the TssRBM-based results were taken from Kivinen and Williams (2012) (including their Multi-Tm: a multiple texture model trained with 256 hidden units). The synthesis results are based on the TSS criterion while the inpainting results are based on MSSIM-scores. In both cases, larger numbers are better.

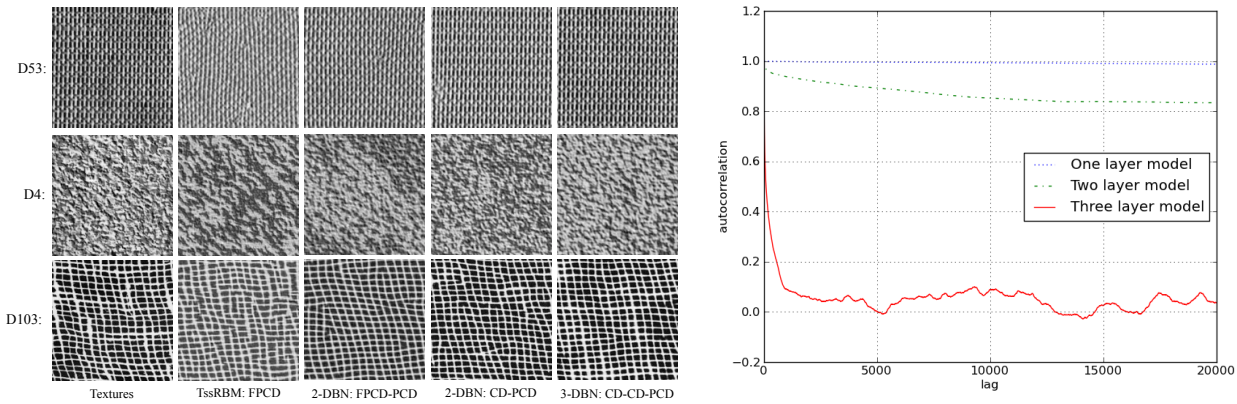

Figure 4: **LEFT:** Synthesized texture D53, D4 and D103 at *full resolution*. The training algorihtms are shown in the layer-order, e.g. 3-DBN: CD-CD-PCD denotes a 3-layer TssDBN trained with CD for the first two layers and with PCD for the upppermost layer. Both depth and the choice of inductive bias have a significant impact on the quality of the model. **RIGHT:** The autocorrelation spectrum of Monte Carlo Markov Chain samples of the texture D103 for our one, two and three-layer models (centered to the mean of the data). All layers are trained with CD, except the uppermost which is trained with PCD (TssRBM trained with FPCD).

sampling iterations in our models while the border was held fixed. The number of inpainting frames was 20 for each texture, and the inpainting were each done with 5 different random seeds, making it a total of 100 inpaintings for each model and each texture. The quality of the inpainting was evaluated using the mean structural similarity index (MSSIM) (Wang *et al.*, 2004) that compares the inpainted region and the ground truth. Fig. 3 compares the texture results of some of the methods under consideration. Table 1 provides the quantitative MSSIM comparison against other similar models. Here again, the TssRBM-based models are competitive with these other probabilistic models of texture.

5.2 Exploring High-Resolution Textures

To further explore the generative power of the DBN models, we move to a more challenging task, specifically, modeling high-resolution textures while keeping the first layer structure unchanged: the same number of filters, the same size (11×11) of the receptive fields and the same size (98×98) of the training patches. This implies that the first layer will face a much more challenging learning task. We show that by adding more hidden layers these difficult tasks are handled very well. In these experiments we consider 1, 2 and 3-layer variants of the TssDBN (with the 1-layer variant being the TssRBM). There are 128 filters with convolutional weight sharing in both of these two layers. Due to the limited space, we only show the results of textures D53, D4 and D103. The other 5 textures yield a similar pattern of results. While the quantitative measures used in the previous experiments are useful to establish an objective comparison between methods, we feel that they are rather imperfect measures of the quality of the texture model and therefore in this section we forgo these measures in favour of simply presenting texture synthesis results for visual inspection. Fig. 4 (right) illustrates the impact of both depth and the inductive bias (FPCD versus CD training) on the training of TssRBM-based models of texture. Overall, we see a clear pattern of improvement with depth. Notably, we also see superior performance with CD-trained lower layers over FPCD-trained lower layers

Depth helps mixing. One key aspect that might help to explain the improvements in the models with depth is that as the models get deeper the mixing rate of the negative phase Gibbs chain improves, as already demonstrated and argued in Bengio *et al.* (2012). Improved mixing of the Gibbs chain improves the performance of training methods such as PCD that rely on it for the estimation of negative phase statistics. It also helps the generation of the samples shown. To demonstrate the improvement in mixing with depth, we assess the mixing rate of three models (one, two and three layer model) trained on D103 via a generalization of autocorrelation to multivariate Markov chains:

$$R(\tau) = \frac{\sum_{t=1}^{N} \hat{v}_t^T \hat{v}_{t+\tau}}{\left[\sum_{t=1}^{N} \hat{v}_t^T \hat{v}_t\right]^{1/2} \left[\sum_{t=1}^{N} \hat{v}_{t+\tau}^T \hat{v}_{t+\tau}\right]^{1/2}}$$

where $\hat{v}_t$ is the image patch sample, v_t, generated at iteration t of Gibbs sampling, centred by the mean, μ_v, of the training data: $\hat{v}_t = v_t - \mu_v$. After training, we run a Markov chain in all of three models and plot the autocorrelation spectrum in Fig. 4 (left). As seen in the figure, mixing becomes relatively very fast in the three-layer model, i.e., samples separated by some distance in the chain are less correlated with each other.

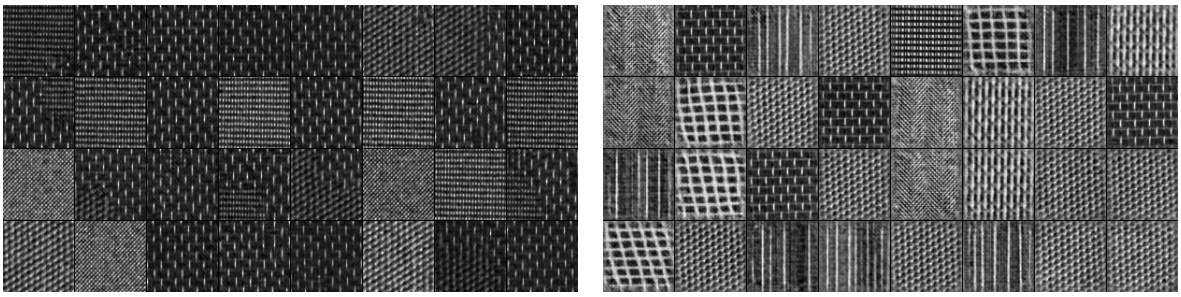

Figure 5: **LEFT:** Multi-texture samples generated by the TssRBM model. **RIGHT:** Multi-texture samples generated by our 3-layer DBN.

CD pretraining vs. PCD and FPCD pretraining. We find that pretraining the lower layers with CD-1 results in better DBN texture models. More specifically, worse results were obtained by PCD pretraining than FPCD pre-training (not shown), and substantially better with CD, as can e.g. be seen in Figure 4 (right). This is consistent with the claims made by Hinton (2012) regarding the advantage of CD vs PCD. It is also consistent with the results of Le Roux and Bengio (2008), which show that maximum likelihood training of the lower layers of a DBN is sub-optimal, and that assuming a high-capacity top layer, the optimal way to train the first layer would be to minimize the KL divergence between the visible units and the stochastic one-step reconstruction, something much closer to what CD does than what PCD does. Another hypothesis is that CD helps here because it makes sure to extract *good features that preserve the input information*, without the constraint that the lower level RBMs do a good job of modeling the density over their inputs (i.e. avoiding spurious modes far from the training samples). Leaving the job of learning an accurate distribution to the top-level RBM, which is trained using PCD – a good approximation to maximum likelihood.

5.3 Learning with Multiple Textures

In this section, we try to further assess the power of the TssDBN by training it on multiple high-resolution heterogeneous textures. We train a 3-layer TssDBN on all 8 textures. The first layer is a TssRBM with 96 filters. The second layer is our new convolutional ssRBM variant introduced in Sec. 4 with 256 filters and receptive fields of size 2×2. The third layer is a convolutional binary RBM with 256 filters and receptive fields of size 2×2. We compare TssDBN with a one layer model (TssRBM with 128 filters). After training, we generate samples from both models and show the results in Fig. 5. We can see that the samples generated by the single layer TssRBM are dominated by those morst regular textures in the training data and fail to capture other more complicate textures. On the other hand, the deep model seems to capture much of the 8 textures that occur in the training set. There are 7 different textures apparent in these 32 samples. We are missing samples of D4, which is a stochastic texture and hard to capture, particularly when most of the training data are highly structured images. Kivinen and Williams (2012) also trained Gaussian RBM with tiled-convolution weight sharing on multiple textures with labels. Training with labels can help the model pick different filters for different textures and thus make the learning problem easier.

6 Conclusions

In this paper, we apply the ssRBM with tiled-convolution weight sharing on texture modeling task. We show that not only is the ssRBM competitive as a single layer model of texture, but that, by being amenable to CD training based on block Gibbs sampling, it is well suited to being incorporated into even more effective deep models of texture. Interestingly, we find that CD training of lower layers yields better models, and that mixing is better in deeper layers. Our integration of the ssRBM into a DBN necessitated the development of a novel RBM with a spike-and-slab visible layer and a binary latent layer. Finally we show our new TssDBN is capable of modeling multiple high-resolution textures when trained in a completely unsupervised fashion.

Acknowledgements

The authors acknowledge the financial support of the NSERC-Ubisoft chair and CIFAR. Special thanks also go to the Theano development team.

References

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Adv. Neural Inf. Proc. Sys. 19*, pages 153–160.

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. (2012). Better mixing via deep representations. Technical Report arXiv:1207.4404, Universite de Montreal.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Courville, A., Bergstra, J., and Bengio, Y. (2011a). A Spike and Slab Restricted Boltzmann Machine. In *AISTATS'2011*.

Courville, A., Bergstra, J., and Bengio, Y. (2011b). Unsupervised models of images by spike-and-slab RBMs. In *ICML'2011*.

Desjardins, G. and Bengio, Y. (2008). Empirical evaluation of convolutional RBMs for vision. Technical Report 1327, Dept. IRO, U. Montréal.

Gregor, K. and LeCun, Y. (2010). Emergence of complex-like cells in a temporal product network with local receptive fields. Technical report, arXiv:1006.0448.

Heess, N., Williams, C. K. I., and Hinton, G. E. (2009). Learning generative texture models with extended fields-of-experts. In *BMVC*.

Hinton, G. E. (2012). Tutorial on deep learning. IPAM Graduate Summer School: Deep Learning, Feature Learning.

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

Kivinen, J. J. and Williams, C. K. I. (2012). Multiple texture Boltzmann machines. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS'2012)*, volume 22 of JMLR: W&CP.

Le, Q., Ngiam, J., Chen, Z., hao Chia, D. J., Koh, P. W., and Ng, A. (2010). Tiled convolutional neural networks. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23 (NIPS'10)*, pages 1279–1287.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient based learning applied to document recognition. *IEEE*, **86**(11), 2278–2324.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In L. Bottou and M. Littman, editors, *ICML 2009*. ACM, Montreal (Qc), Canada.

Lin, W.-C., Hays, J. H., Wu, C., Kwatra, V., and Liu, Y. (2006). Quantitative evaluation on near regular texture synthesis. In *Computer Vision and Pattern Recognition Conference (CVPR '06)*, volume 1, pages 427 – 434.

Neal, R. M. (1994). *Bayesian Learning for Neural Networks*. Ph.D. thesis, Dept. of Computer Science, University of Toronto.

Ranzato, M. and Hinton, G. H. (2010). Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'10)*, pages 2551–2558. IEEE Press.

Ranzato, M., Mnih, V., and Hinton, G. (2010a). Generating more realistic images using gated MRF's. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23 (NIPS'10)*, pages 2002–2010.

Ranzato, M., Mnih, V., and Hinton, G. (2010b). Generating more realistic images using gated MRF's. In *NIPS'2010*.

Ranzato, M., Susskind, J., Mnih, V., and Hinton, G. E. (2011). On deep generative models with applications to recognition. In *CVPR'11*, pages 2857–2864.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *ICML 2008*, pages 1064–1071. ACM.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, **13**(4), 600–612.

Wei, L.-Y., Lefebvre, S., Kwatra, V., and Turk, G. (2009). State of the art in example-based texture synthesis. *Eurographics'09 State of the Art Reports*.

Welling, M., Hinton, G. E., and Osindero, S. (2003). Learning sparse topographic representations with products of Student-t distributions. In *NIPS'2002*.

Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In *NIPS'04*, volume 17, Cambridge, MA. MIT Press.

Younes, L. (1999). On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics and Stochastic Reports*, **65**(3), 177–228.

Zhu, S. C., Liu, X. W., and Wu, Y. N. (2000). Exploring texture ensembles by efficient Markov chain Monte-Carlo - towards a "trichromacy" theory of texture. *IEEE Trans. PAMI*, **22**(6), 554–569.