Multiple Texture Boltzmann Machines

Jyri J. Kivinen **Christopher K. I. Williams**
Institute for Adaptive and Neural Computation
School of Informatics, University of Edinburgh, UK

Abstract

We assess the generative power of the mPoT-model of [10] with tiled-convolutional weight sharing as a model for visual textures by specifically training on this task, evaluating model performance on texture synthesis and inpainting tasks using quantitative metrics. We also analyze the relative importance of the mean and covariance parts of the mPoT model by comparing its performance to those of its subcomponents, tiled-convolutional versions of the PoT/FoE and Gaussian-Bernoulli restricted Boltzmann machine (GB-RBM). Our results suggest that while state-of-the-art or better performance can be achieved using the mPoT, similar performance can be achieved with the mean-only model. We then develop a model for multiple textures based on the GB-RBM, using a shared set of weights but texture-specific hidden unit biases. We show comparable performance of the multiple texture model to individually trained texture models.

1 Introduction

We consider the statistical modelling of natural images using Boltzmann machines. Such Markov random field models with hidden units have shown significant promise for various unsupervised learning problems, including as effective models for the statistics of natural images. One of the most flexible models proposed in the literature is the recent Product of Student-t Experts (PoT) with non-zero means (mPoT) [10], which includes third-order interactions between visible and hidden units. Visual analysis of the samples drawn from the model (with tiled-convolutional weight sharing) as shown in [10, Figure 3] suggests that although piecewise smooth segments with clear intensity discontinuities at their borders can be generated, the model (as trained on natural image patches) does not hallucinate textured regions. The flexibility of the model can be increased by adding additional layers of hidden units, but the samples shown in [11, Figure 3] again do not show textured regions. These results suggest it is over-optimistic to expect a single mPoT model to be able to generate the wide variety of textures seen in natural scenes. However, such a model could be effective for a sufficiently small subproblem such as modelling individual textures. The generation of visual texture is a necessary sub-component of any credible model for visual scenes.

Our paper has two main contributions. First, we assess the power of the mPoT as a model for textures by specifically training on this task. A key advantage of the texture task is that one can assess the generative performance directly e.g. using the texture similarity score from [4]. In contrast, current quantitative assessment of generic natural image models is typically based on discriminative performance of a classifier using features derived from the models, which is a very indirect way of evaluating generative performance. As well as assessing the texture modelling power of the mPoT with tiled-convolutional weight sharing, we also analyze the relative contributions of the mean and covariance parts of the mPoT by comparing its performance to those of its subcomponents, tiled-convolutional versions of the PoT/FoE and the Gaussian-Bernoulli restricted Boltzmann machine (GRBM).

Secondly, we develop a Boltzmann machine which is able to generate multiple textures; a natural extension of a model for specific textures. The model modulates a set of parameters shared across multiple textures with texture-specific parameters to create appropriate texture features. We compare the multi-texture model to single-texture models for constrained and unconstrained texture synthesis.

We begin in Section 2 by reviewing the models con-

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume 22 of JMLR: W&CP 22.

sidered for modelling individual textures. We then describe the experimental setup for the analysis of these models, including data, assessment methods, modelling and inference details in Section 3. This section also gives the results, where we evaluate and analyze the performance and suitability of the methods as models of textures. Section 4 then develops a multi-texture Boltzmann machine and analyzes its properties, including a comparison of the generative performance of the model to those of single-texture models. Section 5 provides a summary and discussion.

2 Modelling of Individual Visual Textures with Boltzmann Machines

Models based on Boltzmann machines have recently shown significant promise for unsupervised learning problems. These models consider two sets of variables, hidden units $\mathbf{h}$, and visible units $\mathbf{v}$, and model the joint distribution of the random variables with the Boltzmann distribution $p(\mathbf{h}, \mathbf{v}) = \frac{1}{Z} \exp\{-\mathrm{E}(\mathbf{v}, \mathbf{h})\}$, where $\mathrm{E}(\mathbf{v}, \mathbf{h})$ is the energy-function defined by the model, and Z is a normalization constant also called the partition function.

In our experiments we consider three models based on Boltzmann machines, all of which model the visible units $\mathbf{v}$ as normally distributed conditional on the hidden units $\mathbf{h}$; this is a setup popular in the modelling of continuous-valued data, such as natural images. Although other models providing such parameterizations do exist, these three models provide the typical spectrum of structure, namely whether the mean is constrained to be zero or not, and whether the covariance matrix is constrained to be diagonal or not. In order to scale up to large images, we use tiled-convolutional weight sharing, as in [10]. In a convolutional weight sharing scheme one considers multiple feature planes each consisting of multiple hidden units in a grid. Each unit connects to a local receptive field of visible units with weights that are feature-plane specific, and shared across the plane. In a tiled-convolutional architecture there are again multiple feature planes which share features, but for a single feature plane the units connect to non-overlapping visible unit receptive fields which together tile the entire image. There are then multiple feature planes associated with similar tilings, but with different offsets.

For simplicity of the presentation, the mathematical descriptions of these models are shown in a non-convolutional setting as much as possible. In the following sections we outline the Gaussian-RBM, PoT and mPoT models, and in section 2.4 we discuss some other generative models for visual texture.

2.1 Gaussian RBM

Convolutional Gaussian RBMs (also called Gaussian-Bernoulli RBMs) were introduced by Lee et. al. [6] and have been popular in modelling large natural images. The energy function for the model for single-channel observables can be defined as follows:

$$\mathrm{E}_{\mathrm{GRBM}}(\mathbf{v}, \mathbf{h}^m) = \frac{(\mathbf{v}-\mathbf{a})^\top(\mathbf{v}-\mathbf{a})}{2\sigma^2} - \sum_k h_k^m \left(b_k + \sigma^{-1}\mathbf{M}_{\cdot k}{}^\top \mathbf{v}\right), \quad (1)$$

where $\mathbf{a}$ denotes the vector of visible-layer biases, b_k denotes a bias and $\mathbf{M}_{\cdot k}$ a weight kernel for feature layer k, and $\sigma > 0$ is a scalar parameter. Conditional on the visible units $\mathbf{v}$, the binary hiddens $\mathbf{h}^m$ are independent and distributed according to a Bernoulli distribution. Conditional on $\mathbf{h}^m$ the visible units have an independent Gaussian distribution: $p(\mathbf{v} \mid \mathbf{h}^m) = \mathcal{N}\left(\mathbf{v}; \mathbf{a} + \sigma\mathbf{M}\mathbf{h}^m, \sigma^2\mathbf{I}\right)$ where σ determines the noise level. It is important to note that the covariance matrix of the conditional distribution is diagonal/spherical (because the units are conditionally independent), and the model places significant focus on modelling the means of these distributions. Below we denote this model Tm, to emphasize that it is tiled-convolutional, and that it focusses on modelling the conditional means, rather than covariances.

2.2 PoT

The Product of Student-t Experts (PoT) [9] model belongs to the Product-of-Experts (PoE) framework, in which the probability of visible units is defined as a normalized product of experts, where each expert is a non-linear potential function acting on visible units. The Fields-of-Experts model (FoE) [12] extends the model to share experts convolutionally across the different unit sites so as to scale up to large images. The PoT can be formulated with auxilliary continuous-valued hidden units $\mathbf{h}^c$ [9], leading to the energy function:

$$\mathrm{E}_{\mathrm{PoT}}(\mathbf{v}, \mathbf{h}^c) = \sum_j h_j^c \left(1 + \frac{1}{2}\left[\mathbf{C}_{\cdot j}{}^\top \mathbf{v}\right]^2\right) + (1-\gamma_j)\log h_j^c, \quad (2)$$

where $\mathbf{C}_{\cdot j}$ is a filter associated with expert j, and γ_j is a scalar parameter. Conditional on the visible units, the hidden units are independent with Gamma distributions. Conditional on the hidden units, the visible units are distributed jointly according to a zero-mean multivariate Gaussian: $p(\mathbf{v} \mid \mathbf{h}^c) = \mathcal{N}\left(\mathbf{v}; \mathbf{0}, \left(\mathbf{C}\,\mathrm{diag}\{\mathbf{h}^c\}\,\mathbf{C}^\top\right)^{-1}\right)$, where $\mathrm{diag}\{\mathbf{h}^c\}$ denotes a diagonal matrix with the elements of $\mathbf{h}^c$ in the diagonal, and $\mathbf{C}$ is a matrix of size the number of

visible units times the number of hidden units (or the experts). In contrast to the Gaussian RBM, the visible units can be coupled in their joint distribution conditional on the hidden units because the precision matrix is not restricted to be diagonal, with the structure being dependent on the hidden unit activation pattern. Below this model is denoted as TPoT, to indicate that it is a tiled-convolutional version of the PoT.

2.3 mPoT

The mPoT [10] is constructed by combining the energy-functions of the GRBM and the PoT, with shared visible units, but different hidden units for the two parts:

$$\mathrm{E}_{\mathrm{mPoT}}(\mathbf{v}, \mathbf{h}^c, \mathbf{h}^m) = \mathrm{E}_{\mathrm{GRBM}}(\mathbf{v}, \mathbf{h}^m) + \mathrm{E}_{\mathrm{PoT}}(\mathbf{v}, \mathbf{h}^c). \tag{3}$$

Conditional on the visible units, the hidden units $\mathbf{h}^c$, and $\mathbf{h}^m$ are independent and distributed according to Gamma-, and Bernoulli-distributions, as in the two models separately. Conditional on the hidden units, the visible units are distributed jointly as multivariate Gaussians: $p(\mathbf{v} \mid \mathbf{h}^c, \mathbf{h}^m) = \mathcal{N}\left(\mathbf{v}; \Lambda\left(\mathbf{a} + \sigma^{-1}\mathbf{M}\mathbf{h}^m\right), \Lambda\right)$, where $\Lambda = \left(\mathbf{C}\mathrm{diag}\left\{\mathbf{h}^c\right\}\mathbf{C}^\top + \sigma^{-2}\mathbf{I}\right)^{-1}$. The mean can be non-zero, and the precision can be non-diagonal, being dependent on the hidden unit assignments. Below this model is denoted as TmPoT, to indicate that it is a tiled-convolutional version of the mPoT.

2.4 Other texture models

Of course texture modelling has a long history and is not restricted to Boltzmann machine models. One simple texture model is a Gaussian random field; for example Heess *et al* [4] consider a simplified FoE with quadratic potentials, which they call the Gaussian FoE (GFoE). The FoE was extended to use bimodal potentials in [4] to create the BiFoE model, and their results show that this generally improved performance over the GFoE and FoE models. In earlier work Zhu *et al* [15] proposed a model based on fixed (rather than learned) filters, but with non-parameteric potentials. Finally we mention the nonparametric texture synthesis method [2]. This grows a patch of texture from a seed, but does so without an explicit generative model; instead it pastes in new pixels based on the match to a reference sample of texture.

The analysis in [4] shows that the FoE with Student-t potentials defines a unimodal density for $p(\mathbf{v})$ with its mode at $\mathbf{v} = \mathbf{0}$, and that the BiFoE creates a multimodal distribution with modes away from the origin. The BiFoE does not have explicit latent variables, although one can reformulate it by replacing the bimodal potential with a mixture of two Gaussians, and adding a hidden unit for each bimodal potential to specify which Gaussian is being used [3, Appendix A.2]. In this case $\mathbf{v}$ is conditionally Gaussian given the hidden units, with a mean that depends on the hidden units, and a fixed non-diagonal covariance. In addition in the mixture-of-Gaussians BiFoE the mean and covariance depend on *the same* filters. The mPoT generalizes this construction by decoupling the mean and covariance parameterization, and providing the freedom for them to vary separately.

3 Dissecting Boltzmann Machine Texture Models

In this section we analyze the performance of the conditionally Gaussian Boltzmann machine models in texture modelling. Below we first discuss the data used for the experiments, and then in section 3.2 give details of how the models were trained. Results for unconstrained texture synthesis are given in section 3.3, and for constrained synthesis (inpainting) in section 3.4.

3.1 Data

The data used in the experiments were Brodatz-texture images[1]. We applied similar rescaling as in [4]: the 640x640 textures were rescaled to either 480x480 or to 320x320, preserving all major texture features. We then normalized each texture to have zero mean, and applied global scaling so that the standard deviation of each texture was 4. This scaling was used because the parameter σ in eq. 1 was fixed to unity in our code; it is equivalent to setting $\sigma = 1/4$ in the GRBM energy (and rescaling $\mathbf{a}$) if the texture were normalized to unit variance. The evaluation metrics used for quantitative analysis are insensitive to these normalization steps. Each image was divided into a top half used for training, and a bottom half for carrying out testing.

3.2 Learning

The training data consisted of patches of size 98×98 randomly cropped out of the the preprocessed training textures, and processed in batches of size 64. We experimented with several receptive field sizes, and the number of hidden units for the models. Increasing the number of hidden units typically improved the generative quality. We used a receptive field size of 11×11, and the tiling was done diagonally with a stride of one pixel. Thus there are 11 sets of filters (one for each offset), and we used 32 filters per set for both the mean

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

$\mathbf{h}^m$ and covariance $\mathbf{h}^c$ hidden units, when applicable. We held $\mathbf{a}$ fixed to zero.

All of the models were trained by approximate maximum likelihood, using stochastic gradient ascent based on Fast Persistent chains Contrastive Divergence (FPCD) [13]. The implementation for training the models heavily used the code by Marc'Aurelio Ranzato[2], especially for the tiled-convolutional mPoT. We will now describe the main details of the learning algorithms: The hidden variables in the TmPoT energy can be integrated out analytically to give the free energy, as in [10, eq. 6]. Parameter learning can be then done by computing the difference of positive and negative phase expectations of the free-energy gradients. In our learning procedure we initialized the negative particles, which are used in the computation of the negative phase, to zeros. They were updated during each iteration using a single-step of hybrid Monte Carlo (HMC) [8], which used a random momentum sampled from a zero-mean, spherical Gaussian, and applied 30 Leapfrog steps.

The models did not have any special boundary units, and therefore at the boundaries and especially at the corners (due to diagonal offsets between the tiles) there were sites which we less constrained than in the center of the image. This often caused boundary artifacts unless care was taken; see supplementary material for details. As in [10], the covariance filters of the TmPoT were pre-multiplied with a whitening transform matrix[3], and during training their L_2-norm was maintained at unity individually by normalization. The normalization avoids the decay of experts to zero, but removes scale adaption, the necessity of which is lessened by the whitening transform.

We initialized the weights $\mathbf{M}$ and $\mathbf{C}$, and also the other parameters in general to small random values. The hidden biases were however initialized to -2. We experimented with various parameter learning rates and combinations for the different models under different textures, but these did not matter much for reasonable ranges of values. For TPoT, we used equal learning rates of 0.001 for $\mathbf{C}$ and γ. For Tm, we used a learning rate of 0.001 for $\mathbf{M}$, and 0.1 for $\mathbf{b}$. We used half of these learning rates for the parameters of the TmPoT models. The learning rates were held fixed for the fast parameters, but annealed for the regular parameters. We also used a small L_1-decay on the weights.

[2] `http://www.cs.toronto.edu/~ranzato/publications/mPoT/mPoT.html`

[3] We used texture-class specific ZCA whitening.

3.3 Unconstrained texture synthesis

Our quantitative analysis of generative performance first considered the quality of texture *samples*. To obtain samples from the models we ran HMC sampling for a large number of iterations, after which their states were stored for analysis. 128 samples of size 120×120 were collected for each model under each texture class. Texture patches and representative model samples (with boundary sites discarded) are shown in Figures 3 and 4(top). Visual inspection shows that while the samples from the TmPoT and Tm are good, TPoT clearly fails to provide a faithful model of the data. Sample filters learned for the different textures using the Tm are shown in the supplementary material Fig. I(bottom).

To provide a quantitative evaluation we compute the Texture Similarity Score (TSS) defined in [4] between each sample and the testing texture patch. For a sample $\mathbf{s}$ and texture image $\mathbf{x}$ the TSS is defined as the maximum of normalized cross correlation (NCC) between them:

$$\mathrm{TSS}(\mathbf{s}, \mathbf{x}) = \max \left\{ \frac{{\mathbf{x}_{(1)}}^\top \mathbf{s}}{\|\mathbf{x}_{(1)}\| \|\mathbf{s}\|}, \ldots, \frac{{\mathbf{x}_{(\mathcal{I})}}^\top \mathbf{s}}{\|\mathbf{x}_{(\mathcal{I})}\| \|\mathbf{s}\|} \right\}, \quad (4)$$

where $\mathbf{x}_{(i)}$ denotes a patch within the image matching the size of $\mathbf{s}$, located at position i, and $\mathcal{I}$ denotes the number of possible patch locations. We used matching window of size 19×19 to compute the score, extracted from a random location in each sample, which is the same size as those of the samples used in [4] to compute their scores. (Note that the results in [4] use the whole texture for both training and texting; in contrast we have a training/test split, see sec. 3.1.)

As in [4], the textures considered for quantitative analysis were D6, D21, D53, and D77 (see top rows of Figure 3). Figure 2 (left) and Table 1 (top) show a summary of quantitative analysis results based on the TSS. The scores for the TmPoT and the Tm are excellent for all the textures, even for the D77, which appears the least homogenous of the textures. While performance of the TmPoT for most textures the highest, it is closely matched by the Tm. The scores for the TPoT are much worse than those of either of the above models; this is consistent with the inability of the FoE model (convolutional PoT) considered in [4] to produce high-quality texture samples. The fact that TPoT doesn't distinguish between $\mathbf{v}$ and $-\mathbf{v}$ is not the only reason for its poor performance in synthesis: scores obtained using absolute values of NCC are still significantly lower for TPoT than for other models (Abs-TSS sample mean$\pm$stds: D6: 0.6273 ± 0.0714, D21: 0.7692 ± 0.0911, D53: 0.7765 ± 0.1039, D77: 0.7055 ± 0.0896).

When comparing these results to Figure 3(a) in [4], note that the filters used there were 7×7, and that 9 sets of filters were used, in a fully convolutional rather than (diagonally) tiled-convolutional fashion. Although many more parameters need to be learned for our models, within each 11×11 block in the image (except for boundaries) there were only 11×32 experts due to the diagonal stride between tiles used in our experiments; this is clearly less than $11 \times 11 \times 9$ experts of [4]. There are also the training/test split differences noted above. With these caveats we note that all mean TSS scores are clearly superior even with the Tm models over BiFoE, and the difference is particularly noticable for D6 and D77.

3.4 Constrained texture synthesis

We used an inpainting evaluation protocol very similar to [4]: We took patches out of the test texture images, and created a square hole inside by setting texture values within the square to zeroes. The inpainting task was then to produce reasonable values to the zeroed out pixels. The quality was measured by the (i) NCC score, (ii) mean structural similarity index (MSSIM) [14] between the inpainted region and the ground truth region, and (iii) TSS between the inpainted region and the test portion of the Brodatz texture. Instead of using 70×70 images, we used 76×76 images and used 54×54 instead of a 50×50 inpainting square as in [4]. The reference frame border was then 11×11, compared to 10×10 of [4]. The inpainting was done for the models by running HMC sampling, during which we constrained the reference border. The number of inpainting frames used in the experiments was 20 for each texture class, and the inpainting was done with 5 different random number generator initial states, producing 100 result images for each model under each texture class. We also compared against the nonparametric method by Efros & Leung (E&L) [2], and our implementation of that method used the training half of the image as the training data. The 'neighbourhood window' for infilling from the training data was 15×15, as used in [4].

Inpainting results are summarized quantitatively w.r.t. NCC in Figure 2 (right) and Table 1 (bottom). Results for MSSIM and TSS are shown in Section C of the supplementary material. Example inpainting results for Tm are shown in Figure 4, and for the other models in the supplementary material (Fig. IV). Our experiments suggest that by providing a reference frame, the models are able to improve the quality of the samples as measured by the TSS[4] over those from texture synthesis[5]. As in the texture synthesis task, the scores for the TmPoT and the Tm model are highest in general, and comparable to each other. Interestingly, providing the reference frame provides a performance boost to the TPoT in relation to the other models: Although it still scores slightly lower than the other models on most textures, its performance is even slightly better than the other models for the D53 texture[6]. Comparing to the Efros & Leung and BiFoE results (last row of Table 1), we observe very similar results for D6, D21 and D53, but that our performance is markedly better for D77.

The following section develops a novel framework for Boltzmann machines to generate multiple textures, where we will use Tm as the base-model, since it obtained state-of-the-art results, and it is computationally much less complicated than the TmPoT[7].

4 Multi-Texture Boltzmann Machines

Current Boltzmann machine models of textures model individual textures. Here we describe a framework for multiple textures. Our model has two sets of parameters Θ; θ_{global} are shared parameters across the different classes, while the parameters $\{\theta_m\}$ are specific to each individual texture class $m = 1, \ldots, M$.

Let $\mathbf{V}_m$ denote the visibles of the images in texture class m. We assume that each of the M probabilities $p(\mathbf{V}_m \mid \theta_m, \theta_{\text{global}})$ are defined by Gaussian RBMs, as defined in equation 1. The weights $\mathbf{M}$ of these models are set to be global, while the biases $\mathbf{b}$ are set to be texture-specific, so that $p(\mathbf{V}_1, \ldots, \mathbf{V}_M \mid \Theta) = \prod_{m=1}^{M} p(\mathbf{V}_m \mid \mathbf{b}_m, \mathbf{M})$. Switching between the different classes is achieved by having a high-level categorical variable y with M states denoting the different textures. The appropriate biases are switched in by selecting the state of y, similar to the implicit mixture construction in [7].

As in the previous experiments, we use tiled-convolutional weight sharing with the model in the following experiments. We denote this model the multi-Tm. To further motivate the multi-Tm, imagine that we start with a single-Tm model for a specific texture, and then add the filters from all the other texture models to the energy, but setting the biases for the filters from all of the models to large negative values. This will have the effect of "turning off" the filters from the other models, leaving in effect the original single-

[4] MSSIM cannot be used for assessing both of the tasks.

[5] The sampling and inpainting scores are not directly comparable because the patch sizes for scoring were different, and typically the smaller the patch the larger the score.

[6] The slightly worse performance of the TmPoT is likely to be related to local optima and boundary issues (which for that model were most problematic).

[7] Boundaries were typically also easier to deal with it.

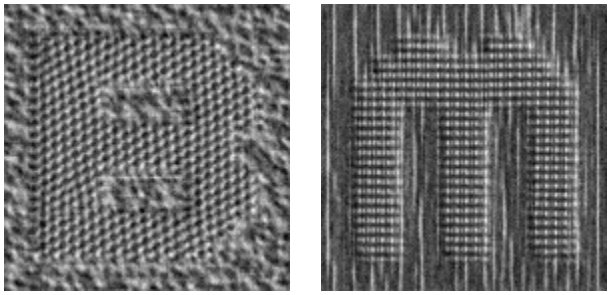

Figure 1: Texture results obtained by using different bias settings for letters and the background.

Tm model. Thus this model can be made to mimic each of the original single-Tm models by adjustment of the biases. However, the real multi-Tm can be more powerful by sharing filters across textures.

4.1 Learning

We learned the parameters of the models by approximate maximum likelihood, using stochastic gradient ascent based on Fast Persistent chains Contrastive Divergence (FPCD), as in above. We assigned sets of 64 negative particles to each of the texture classes, which were updated with HMC sampling, so that for texture class m samples at an epoch were drawn from the model specified by current parameters $\mathbf{b}_m$ and weights $\mathbf{M}$, using similar techniques as described before. Our implementation loops over texture classes updating their negative particles which are then used to compute the gradients and parameter updates for their class-specific biases, and accumulates gradients w.r.t weights which are used to update the weights once all classes have been sweeped/statistics collected.

4.2 Experiments

We have trained models for 8 Brodatz texture categories (D6, D21, D53, D77, D4, D16, D68, and D103) with 96, 128, and 256 features for each tile set, shared over all of the textures; the models for individual textures (single-Tms) each have 32 specific features, totalling $32 \times 8 = 256$ features. Examples of the filters learned and analysis of their specificity to various textures are discussed in the supplementary material.

We have evaluated the sampling and inpainting performance of the models using the setup of previous section. Representative samples from the model with 128 features per site are shown in Figure 3 (bottom row), and Figure 4 (bottom rows in the top (synthesis) and bottom (inpainting) blocks). Visual inspection of the figures shows that they are comparable to those of the individually trained Tm-models, and that the models can capture the statistics of a wide variety of textures effectively. Numerically the performances are also similar, as can be seen from Figure 2 and Table 1, with the exception of higher synthesis performance for multi-Tm models on D53, and higher NCC inpainting performance for the individually trained D77. Table 1 shows that the performance of the multi-Tm in general improves as the number of features is increased.

We have also experimented with varying the biases in a spatial fashion. In Fig. 1 bias settings corresponding to two different textures have been used for letters and the background; the model transitions nicely between the two textures.

5 Summary and Discussion

We have analyzed the generative power of mPoT and its subcomponents for the task of single texture modelling. Our results show that it is essential to not restrict the conditional mean of visible units to be zero, consistent with previous findings in [4]. The results on the texture synthesis and inpainting tasks are generally as good as and sometimes better than the start-of-the-art results in [4]. Our results show that for this task it is the mean hidden units that are much more important, especially in unconstrained sampling. In future work we would like to investigate some further ideas for dealing with boundary effects, e.g. specific boundary feature sets; see also discussion in [5].

We have also developed Boltzmann machines capable of modelling multiple textures, and applied it to the tiled-convolutional Gaussian RBM. By considering a shared set of weights but texture-specific hidden unit biases, we have shown comparable performance to the individually-trained texture models which already provide state-of-the-art results. The feature sharing by multi-texture models is expected to yield savings in terms of the number of features needed to model several categories, and provides a natural route for extension to a more comprehensive natural image model. In this paper we focused on images containing single textures. We are currently working on extending the model to be able to switch between generating differently textured regions, and developing an image segmentation model. In the deep belief network framework this can be done by letting the biases $\mathbf{b}$ in E_{GRBM} depend on a higher layer of hidden units.

Acknowledgements: We thank Nicolas Heess for helpful discussions, and the anonymous referees who helped improve the paper. This work is supported in part by the IST Programme of the EC under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

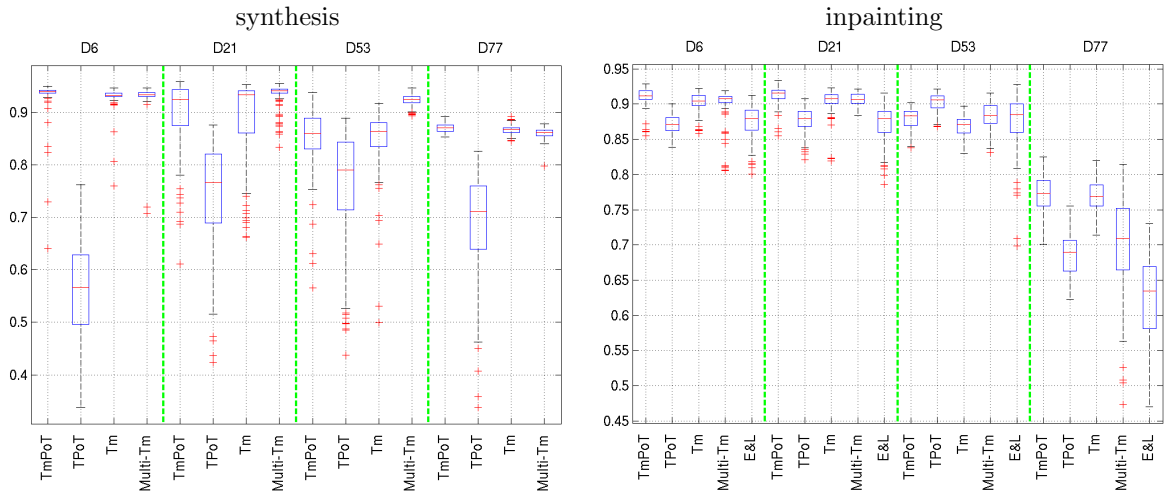

Figure 2: Quality assessment for the models, based on TSS/NCC between sample patches/inpainted area and corresponding Brodatz texture for unconstrained (left) and constrained (right) synthesis. The Multi-Tm model has 256 features per site. Boxes indicate the upper and lower quartiles as well as the median (red bar) of the TSS/NCC distributions; whiskers show extent of the rest of the data; red crosses denote outliers.

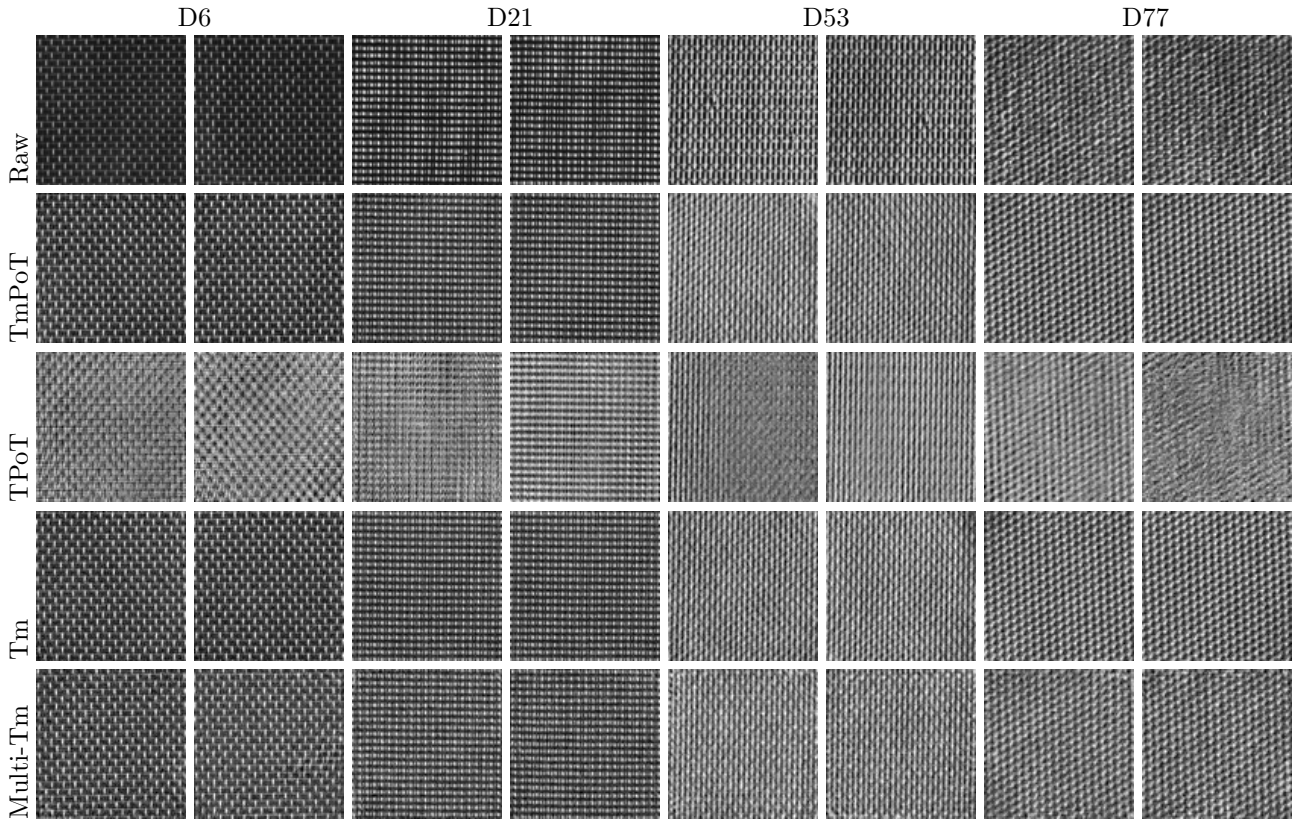

Figure 3: Example data patches (top row), and model samples (other rows), with each case scaled independently to cover the full intensity range. The Multi-Tm model has 128 features per site.

Synthesis	D6	D21	D53	D77
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
Tm	0.9301 ± 0.0207	0.8901 ± 0.0792	0.8485 ± 0.0606	0.8663 ± 0.0084
Multi-Tm (96)	0.8038 ± 0.1344	0.8800 ± 0.0533	0.8610 ± 0.0586	0.8175 ± 0.0394
Multi-Tm (128)	0.8890 ± 0.0821	0.9067 ± 0.0319	0.8881 ± 0.0462	0.8326 ± 0.0235
Multi-Tm (256)	0.9304 ± 0.0280	0.9346 ± 0.0205	0.9231 ± 0.0103	0.8610 ± 0.0096
Bi-FoE	0.7573 ± 0.0594	0.8710 ± 0.0317	0.8266 ± 0.0869	0.6464 ± 0.0215
Inpainting	D6	D21	D53	D77
TmPoT	0.9106 ± 0.0138	0.9127 ± 0.0128	0.8782 ± 0.0166	0.7735 ± 0.0273
TPoT	0.8711 ± 0.0130	0.8764 ± 0.0176	0.9028 ± 0.0125	0.6859 ± 0.0290
Tm	0.9029 ± 0.0135	0.9039 ± 0.0179	0.8679 ± 0.0162	0.7709 ± 0.0245
Multi-Tm (96)	0.8773 ± 0.0202	0.8879 ± 0.0090	0.8537 ± 0.0172	0.7097 ± 0.0402
Multi-Tm (128)	0.8891 ± 0.0203	0.8948 ± 0.0101	0.8701 ± 0.0195	0.7124 ± 0.0488
Multi-Tm (256)	0.8997 ± 0.0246	0.9068 ± 0.0095	0.8826 ± 0.0208	0.7032 ± 0.0725
Efros&Leung	0.8746 ± 0.0239	0.8724 ± 0.0262	0.8732 ± 0.0412	0.6211 ± 0.0582
Bi-FoE [4]	0.8769 ± 0.0163	0.8653 ± 0.0244	0.9145 ± 0.0125	0.6567 ± 0.0205

Table 1: Sample means and standard deviations of the texture synthesis (top) TSS- and inpainting (bottom) NCC-scores. We thank Nicolas Heess for providing the Bi-FoE results for the synthesis task. The inpainting results for Bi-FoE [4] are shown for rough comparison/indicative purposes, as they were obtained using a slightly different experimental setup. See supplementary material for the inpainting results w.r.t MSSIM and TSS.

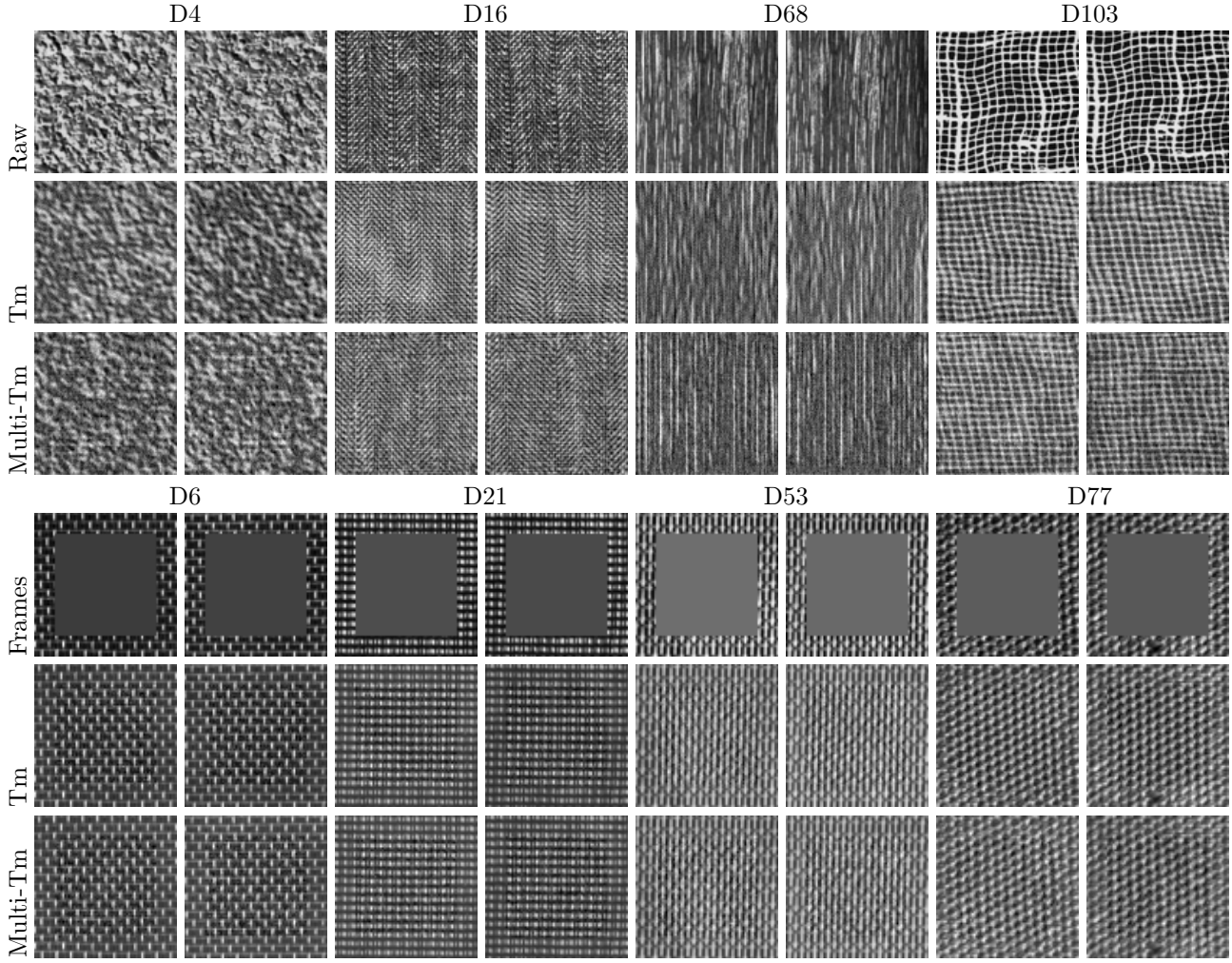

Figure 4: Synthesis (top block) and inpainting (bottom block) results. Example data patches/inpainting frames (top row), and representative results for Tm-models (middle row) and a 128-feature multi-Tm (bottom row).

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[2] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc International Conference on Computer Vision (ICCV 1999)*, pages 1033–1038. 1999.

[3] N. Heess. *Learning generative models of mid-level structure in natural images*. PhD thesis, School of Informatics, University of Edinburgh, 2011.

[4] N. Heess, C.K.I. Williams, and G.E. Hinton. Learning generative texture models with extended Fields-of-Experts. In *BMVC*, 2009.

[5] A. Krizhevsky. Convolutional deep belief networks on CIFAR-10. Technical report, Dept of Computer Science, University of Toronto, 2010.

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

[7] V. Nair and G. E. Hinton. Implicit mixtures of restricted Boltzmann machines. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1145–1152. 2009.

[8] Neal, R. M. *Bayesian Learning for Neural Networks*. Springer, New York, 1996. Lecture Notes in Statistics 118.

[9] S. Osindero, M. Welling, and G. E. Hinton. Topographic product models applied to natural scene statistics. *Neural Computation*, 18(2):381–414, 2006.

[10] M. Ranzato, V. Mnih, and G. E. Hinton. Generating more realistic images using gated MRF's. In *NIPS 23*. 2010.

[11] M. Ranzato, J. Susskind, V. Mnih, and G. E. Hinton. On deep generative models with applications to recognition. In *CVPR*. 2011.

[12] S. Roth and M. J Black. Fields of Experts: A framework for learning image priors. In *CVPR*, pages II: 860–867, 2005.

[13] T. Tieleman and G.E. Hinton. Using fast weights to improve Persistent Contrastive Divergence. In *Proceedings of the 26th International Conference on Machine learning*, pages 1033–1040. ACM New York, NY, USA, 2009.

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[15] S-C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *Int. J. Comput. Vision*, 27(2):107–126, 1998.