WHAT DOES IT TAKE TO GENERATE NATURAL TEXTURES?

Ivan Ustyuzhaninov[*,1,2,3], Wieland Brendel[*,1,2], Leon Gatys[1,2,3], Matthias Bethge[1,2,3,4]

[*]contributed equally
[1]Centre for Integrative Neuroscience, University of Tübingen, Germany
[2]Bernstein Center for Computational Neuroscience, Tübingen, Germany
[3]Graduate School of Neural Information Processing, University of Tübingen, Germany
[4]Max Planck Institute for Biological Cybernetics, Tübingen, Germany

first.last@bethgelab.org

ABSTRACT

Natural image generation is currently one of the most actively explored fields in Deep Learning. Many approaches, e.g. for state-of-the-art artistic style transfer or natural texture synthesis, rely on the statistics of hierarchical representations in supervisedly trained deep neural networks. It is, however, unclear what aspects of this feature representation are crucial for natural image generation: is it the depth, the pooling or the training of the features on natural images? We here address this question for the task of natural texture synthesis and show that none of the above aspects are indispensable. Instead, we demonstrate that natural textures of high perceptual quality can be generated from networks with only a single layer, no pooling and random filters.

1 INTRODUCTION

During the last two years several different approaches towards natural image generation have been suggested, among them generative adversarial networks (Goodfellow et al., 2014; Chen et al., 2016), probabilistic generative models like the conditional PixelCNN (van den Oord et al., 2016b;a) or maximum entropy models that rely on the representations of deep neural networks (e.g. Gatys et al., 2015b; Johnson et al., 2016; Ulyanov et al., 2016). The latter approach has been particularly groundbreaking for artistic style transfer and natural texture generation (e.g. Gatys et al., 2015a;b) and has the potential to uncover the regularities that supervisedly trained deep neural networks infer from natural images.

For the sake of clarity and concreteness, this paper will focus on natural texture synthesis. Parametric texture models aim to uniquely describe each texture by a set of statistical measurements that are taken over the spatial extent of the image. Each image with the same spatial summary statistics should be perceived as the same texture. Consequently, synthesizing a texture corresponds to finding a new image that reproduces the summary statistics inferred from the reference texture. Starting from Nth-order joint histograms of the pixels by Julesz (1962), many different statistical measures have been proposed (see e.g. Heeger & Bergen, 1995; Portilla & Simoncelli, 2000). The quality of the synthesized textures is usually determined by human inspection; the synthesis is successful if a human observer cannot tell the reference texture from the synthesized ones.

The current state of the art in parametric texture modeling (Gatys et al., 2015a) employs the hierarchical image representation in a deep 19-layer convolutional network (Simonyan & Zisserman (2014); in the following referred to as VGG network) that was trained on object recognition in natural images(Russakovsky et al. (2015)). In this model textures are described by the raw correlations between feature activations in response to the texture image from a collection of network layers (see section 5 for details). Since its initial reception several papers explored which additional elements or constraints can further increase the perceptual quality of the generated textures (Berger & Memisevic, 2016; Liu et al., 2016; Aittala et al., 2016). In this work we go the opposite way and ask which elements of the original texture synthesis algorithm (Gatys et al., 2015a) are absolutely indispensable.

In particular two aspects have been deemed critical for natural texture synthesis: the hierarchical multi-layer representation of the textures, and the supervised training of the feature spaces. Here we show that neither aspect is imperative for texture modeling and that in fact a single convolutional layer with random features can synthesize textures that often rival the perceptual quality of Gatys et al. (2015a). This is in contrast to earlier reports (Gatys et al., 2015a) that suggested that networks with random weights fail to generate perceptually interesting images. We suggest that this discrepancy originates from a more elaborate tuning of the optimization procedure (see section 4).

Our main contributions are:

- We present a strong minimal baseline for parametric texture synthesis that solely relies on a single-layer network and random, data-independent filters.
- We show that textures synthesized from the baseline are of high quality and often rival state-of-the-art approaches, suggesting that the depth and the pre-training of multi-layer image representations are not as indispensable for natural image generation as has previously been thought.
- We test and compare a wide range of single-layer architectures with different filter-sizes and different types of filters (random, hand-crafted and unsupervisedly learnt filters) against the state-of-the-art texture model by Gatys et al. (2015a).
- We utilize a quantitative texture quality measure based on the synthesis loss in the VGG-based model (Gatys et al., 2015a) to replace the common-place evaluation of texture models through qualitative human inspection.
- We discuss a formal generalization of maximum entropy models to account for the natural variability of textures with limited spatial extent.

2 Convolutional Neural Network

If not mentioned otherwise, all our models employ single-layer CNNs with standard rectified linear units (ReLUs) and convolutions with stride one, no bias and padding $(f-1)/2$ where f is the filter-size. This choice ensures that the spatial dimension of the output feature maps is the same as the input. All networks except the last one employ filters of size $11 \times 11 \times 3$ (filter width $\times$ filter height $\times$ no. of input channels), but the number of feature maps as well as the selection of the filters differ:

- **Fourier-363:** Each color channel (R, G, B) is filtered separately by each element $\mathbf{B}_i \in \mathbb{R}^{11\times 11}$ of the 2D Fourier basis ($11 \times 11 = 121$ feature maps/channel), yielding $3 \cdot 121 = 363$ feature maps in total. More concretely, each filter can be described as the tensor product $\mathbf{B}_i \otimes \mathbf{e}_k$ where the elements of the unit-norm $\mathbf{e}_k \in \mathbb{R}^3$ are all zero except one.
- **Fourier-3267:** All color channels (R, G, B) are filtered simultaneously by each element $\mathbf{B}_i$ of the 2D Fourier basis but with different weighting terms $w_R, w_G, w_B \in [1, 0, -1]$, yielding $3 \cdot 3 \cdot 3 \cdot 121 = 3267$ feature maps in total. More concretely, each filter can be described by the tensor product $\mathbf{B}_i \otimes [w_R, w_G, w_B]$.
- **Kmeans-363:** We randomly sample and whiten 1e7 patches of size 11×11 from the Imagenet dataset (Russakovsky et al., 2015), partition the patches into 363 clusters using k-means (Rubinstein et al., 2009), and use the cluster means as convolutional filters.
- **Kmeans-3267:** Same as Kmeans-363 but with 3267 clusters.
- **Kmeans-NonWhite-363/3267:** Same as Kmeans-363/3267 but without whitening of the patches.
- **Kmeans-Sample-363/3267:** Same as Kmeans-363/3267, but patches are only sampled from the target texture.
- **PCA-363:** We randomly sample 1e7 patches of size 11×11 from the Imagenet dataset (Russakovsky et al., 2015), vectorize each patch, perform PCA and use the set of principal axes as convolutional filters.
- **Random-363:** Filters are drawn from a uniform distribution according to (Glorot & Bengio, 2010), 363 feature maps in total.
- **Random-3267:** Same as Random-363 but with 3267 feature maps.

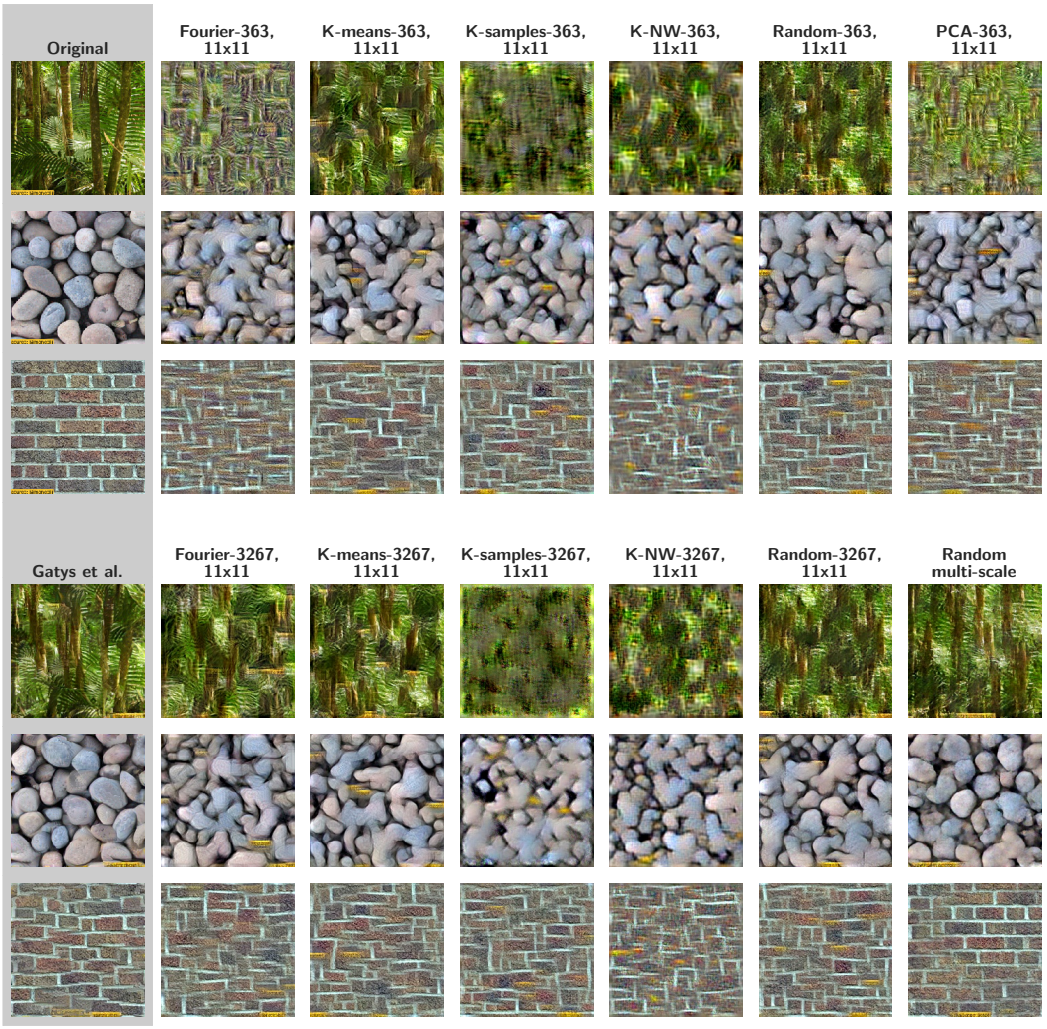

Figure 1: Influence of the feature maps on texture synthesis performance. (Top) Samples synthesized from several single-layer models with 363 feature maps (see sec. 2) for three different textures (rows). Reference textures are shown in the first column. (Bottom) Samples synthesized from several single-layer models with 3267 feature maps (see sec. 2) for three different textures (rows). Additionally, the first column shows samples from the VGG model (Gatys et al., 2015a), and the last column from the multi-scale model (with 1024 feature maps).

- **Random-Multiscale** Eight different filter sizes $f \times f \times 3$ with $f = 3, 5, 7, 11, 15, 23, 37, 55$ and 128 feature maps each (1024 feature maps in total). Filters are drawn from a uniform distribution according to (Glorot & Bengio, 2010).

The networks were implemented in Lasagne (Dieleman et al., 2015; Theano Development Team, 2016). We remove the DC component of the inputs by subtracting the mean intensity in each color channel (estimated over the Imagenet dataset (Russakovsky et al., 2015)).

3 Texture Model

The texture model closely follows (Gatys et al., 2015a). In essence, to characterise a given vectorised texture $\mathbf{x} \in \mathbb{R}^M$, we first pass $\mathbf{x}$ through the convolutional layer and compute the output activations. The output can be understood as a non-linear filter bank, and thus its activations form a set of filtered images (so-called feature maps). For N distinct feature maps, the rectified output activations can be

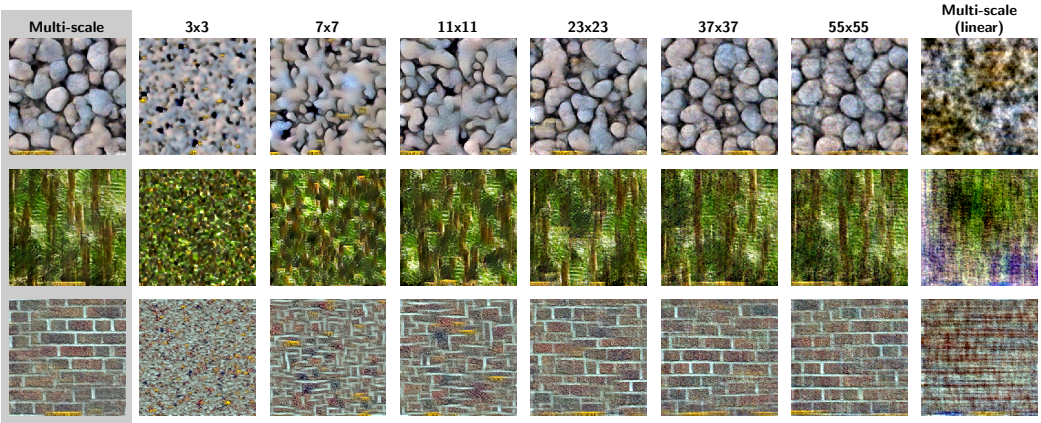

Figure 2: Influence of the scale and the non-linearity on texture synthesis performance. (1st column) Samples from the random multi-scale model for comparison (same as in Fig. 1). (2nd - 7th colum) Samples from the random single-scale model with different spatial filter sizes. (Last column) Samples from the random multi-scale model without ReLU nonlinearity.

described by a matrix $\mathbf{F} \in \mathbb{R}^{N \times M}$. To capture the stationary structure of the textures, we compute the covariances (or, more precisely, the Gramian matrix) $\mathbf{G} \in \mathbb{R}^{N \times N}$ between the feature activations $\mathbf{F}$ by averaging the outer product of the point-wise feature vectors,

$$G_{ij} = \frac{1}{M} \sum_{m=1}^{M} F_{im} F_{jm}. \tag{1}$$

We will denote $\mathbf{G}(\mathbf{x})$ as the Gram matrix of the feature activations for the input $\mathbf{x}$. To determine the relative distance between two textures $\mathbf{x}$ and $\mathbf{y}$ we compute the euclidean distance of the normalized Gram matrices,

$$d(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{\sum_{m,n} G_{mn}(\mathbf{x})^2} \sqrt{\sum_{m,n} G_{mn}(\mathbf{y})^2}} \sum_{i,j=1}^{N} \left(G_{ij}(\mathbf{x}) - G_{ij}(\mathbf{y}) \right)^2. \tag{2}$$

To compare with the distance in the raw pixel values, we compute

$$d_p(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{\sum_{m} x_m^2} \sqrt{\sum_{m} y_m^2}} \sum_{i=1}^{N} \left(x_i - y_i \right)^2. \tag{3}$$

4 Texture Synthesis

To generate a new texture we start from a uniform noise image (in the range [0, 1]) and iteratively optimize it to match the Gram matrix of the reference texture. More precisely, let $\mathbf{G}(\mathbf{x})$ be the Gram matrix of the reference texture. The goal is to find a synthesised image $\tilde{\mathbf{x}}$ such that the squared distance between $\mathbf{G}(\mathbf{x})$ and the Gram matrix $\mathbf{G}(\tilde{\mathbf{x}})$ of the synthesized image is minimized, i.e.

$$\tilde{\mathbf{x}} = \underset{\mathbf{y} \in \mathbb{R}^M}{\operatorname{argmin}} \; E(\mathbf{y}), \tag{4}$$

$$E(\mathbf{y}) = \frac{1}{\sum_{i,j=1}^{N} G_{ij}(\mathbf{x})^2} \sum_{i,j=1}^{N} \Big(G_{ij}(\mathbf{x}) - G_{ij}(\mathbf{y}) \Big)^2. \tag{5}$$

The gradient $\partial E(\mathbf{y}) / \partial \mathbf{y}$ of the reconstruction error with respect to the image can readily be computed using standard backpropagation, which we then use in conjunction with the L-BFGS-B algorithm (Jones et al., 2001–) to solve (4). We leave all parameters of the optimization algorithm at their default value except for the maximum number of iterations (2000), and add a box constraints with range [0, 1]. In addition, we scale the loss and the gradients by a factor of 10^7 in order to avoid early stopping of the optimization algorithm.

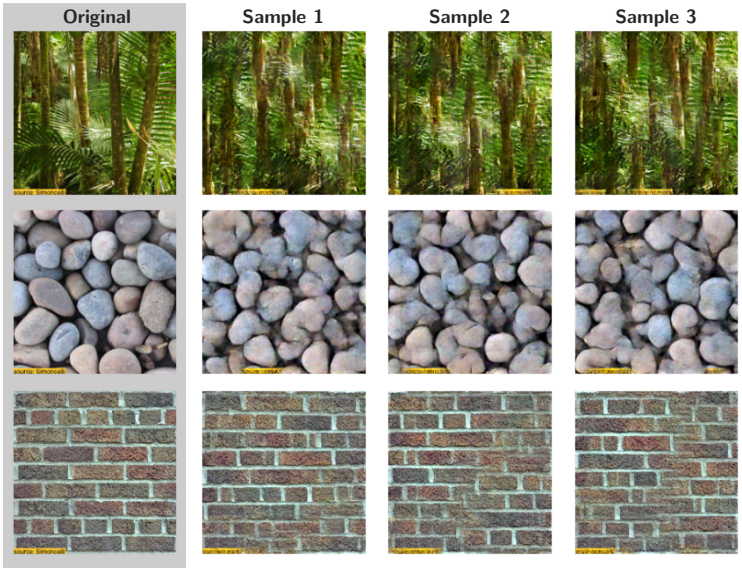

Figure 3: Each row shows the reference texture (left, gray background) and three samples that were synthesized from different (random) initial images using our multi-scale model. Most importantly, the multi-scale model generates samples that are perceptually different. All three textures are taken from Portilla & Simoncelli (2000).

5 Texture Evaluation

Evaluating the quality of the synthesized textures is traditionally performed by human inspection. Optimal texture synthesis should generate samples that humans perceive as being the same texture as the reference. The high quality of the synthesized textures by (Gatys et al., 2015a) suggests that the summary statistics from multiple layers of VGG can approximate the perceptual metric of humans. Even though the VGG texture representation is not perfect, this allows us to utilize these statistics as a more objective quantification of texture quality.

For all details of the VGG-based texture model see (Gatys et al., 2015a). Here we use the standard 19-layer VGG network (Simonyan & Zisserman, 2014) with pretrained weights and average- instead of max-pooling[1]. We compute a Gram matrix on the output of each convolutional layer that follows a pooling layer. Let $\mathbf{G}^\ell(.)$ be the Gram matrix on the activations of the ℓ-th layer and

$$E^\ell(\mathbf{y}) = \frac{1}{\sum_{i,j=1}^{N} G^\ell_{ij}(\mathbf{x})^2} \sum_{i,j=1}^{N} \Big(G^\ell_{ij}(\mathbf{x}) - G^\ell_{ij}(\mathbf{y}) \Big)^2. \tag{6}$$

the corresponding relative reconstruction cost. The total reconstruction cost is then defined as the average distance between the reference Gram matrices and the synthesized ones, i.e.

$$E(\mathbf{y}) = \frac{1}{5} \sum_{\ell=1}^{5} E^\ell(\mathbf{y}). \tag{7}$$

This cost is reported on top of each synthesised texture in Figures 4. To visually evaluate samples from our single- and multi-scale model against the VGG-based model (Gatys et al., 2015a), we additionally synthesize textures from VGG by minimizing (7) using L-BFGS-B as in section 4.

6 Results

In Fig. 1 we show textures synthesised from two random single- and multi-scale models, as well as eight other non-random single-layer models for three different source images (top left). For

[1] https://github.com/Lasagne/Recipes/blob/master/modelzoo/vgg19.py as accessed on 12.05.2016.

comparison, we also plot samples generated from the VGG model by Gatys et al. (Gatys et al., 2015a) (bottom left). There are roughly two groups of models: those with a small number of feature maps (363, top row), and those with a large number of feature maps (3267, bottom row). Only the multi-scale model employs 1024 feature maps. Within each group, we can differentiate models for which the filters are unsupervisedly trained on natural images (e.g. sparse coding filters from k-means), principally devised filter banks (e.g. 2D Fourier basis) and completely random filters (see sec. 2 for all details). All single-layer networks, except for multi-scale, feature 11×11 filters. Remarkably, despite the small spatial size of the filters, all models capture much of the small- and mid-scale structure of the textures, in particular if the number of feature maps is large. Notably, the scale of these structures extends far beyond the receptive fields of the single units (see e.g. the pebble texture). We further observe that a larger number of feature maps generally increases the perceptual quality of the generated textures. Surprisingly, however, completely random filters perform on par or better then filters that have been trained on the statistics of natural images. This is particularly true for the multi-scale model that clearly outperforms the single-scale models on all textures. The captured structures in the multi-scale model are generally much larger and often reach the full size of the texture (see e.g. the wall).

While the above results show that for natural texture synthesis one neither needs a hierarchical deep network architecture with spatial pooling nor filters that are adapted to the statistics of natural images, we now focus on the aspects that are crucial for high quality texture synthesis. First, we evaluate whether the success of the random multi-scale network arises from the combination of filters on multiple scales or whether it is simply the increased size of its largest receptive fields (55×55 vs. 11×11) that leads to the improvement compared to the single-scale model. Thus, to investigate the influence of the spatial extend of the filters and the importance of combining multiple filter sizes in one model, we generate textures from multiple single-scale models, where each model has the same number of random filters as the multi-scale model (1024) but only uses filters from a single scale of the multi-scale model (Fig. 2). We find that while 3×3 filters mainly capture the marginal distribution of the color channels, larger filters like 11×11 model small- to mid-scale structures (like small stones) but miss more long-range structures (larger stones are not well separated). Very large filters like 55×55, on the other hand, are capable of modeling long-range structures but then miss much of the small- to midscale statistics (like the texture of the stone). Therefore we conclude that the combination of different scales in the multi-scale network is important for good texture synthesis since it allows to simultaneously model small-, mid- and long-range correlations of the textures. Finally we note that a further indispensable component for good texture models are the non-linearities: textures synthesised the multi-scale model without ReLU (Fig. 2, right column) are unable to capture the statistical dependencies of the texture.

The perceptual quality of the textures generated from models with only a single layer and random filters is quite remarkable and surpasses parametric methods like Portilla & Simoncelli (2000) that have been state-of-the-art two years ago (before the use of DNNs). The multi-scale model often rivals the current state of the art (Gatys et al., 2015a) as we show in Fig. 4 where we compare samples synthesized from 20 different textures for the random single- and multi-scale model, as well as VGG. The multi-scale model generates very competitive samples in particular for textures with extremely regular structures across the whole image (e.g. for the brick wall, the grids or the scales). In part, this effect can be attributed to the more robust optimization of the single-layer model that is less prone to local minima then the optimization in deeper models. This can be seen by initializing the VGG-based synthesis with textures from the single-layer model, which consistently yields superior synthesis results (see Appendix A, Fig. 5). In addition, for a few textures such as the grid structures, the VGG-based loss is paradoxically lower for samples from the multi-scale model then for the VGG-based model (which directly optimized the VGG-based loss). This suggests that the naive synthesis performed here favors images that are perceptually similar to the reference texture and thus looses variability (see sec. 7 for further discussion). Nonetheless, samples from the single-layer model still exhibit large perceptual differences, see Fig. 3. The VGG-based loss (7) appears to generally be an acceptable approximation of the perceptual differences between the reference and the synthesized texture. Only for a few textures, especially those with very regular men-made structures (e.g. the wall or the grids), the VGG-based loss fails to capture the perceptual advantage of the multi-scale synthesis.

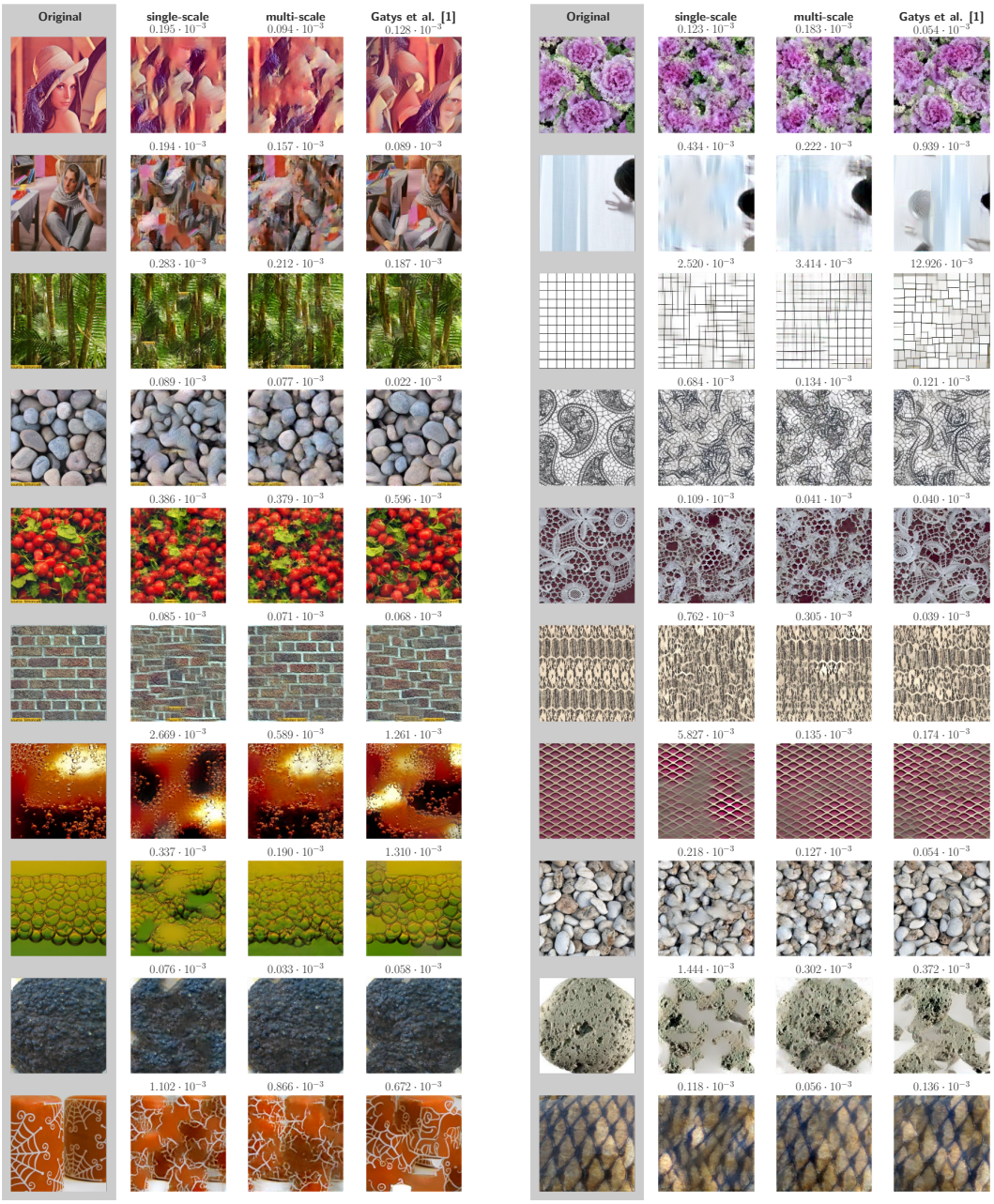

Figure 4: Each row shows the reference texture (left, gray background) and three samples that were synthesized from different (random) initial images using three different models: single-layer network with 1024 feature maps and random 11x11 filters; multi-scale single layer network with filters of sizes $f \times f$, where $f = \{3, 5, 7, 11, 15, 23, 37, 55\}$ and 128 feature maps correspond to filters of each size; and the VGG-based model (Gatys et al., 2015a). Numbers above figures show the values of the normalized VGG-loss (7) for corresponding textures.

7 DISCUSSION

We proposed a generative model of natural textures based on a single-layer convolutional neural network with completely random filters and showed that the model is able to qualitatively capture the perceptual differences between natural textures. Samples from the model often rival the current state-of-the-art (Gatys et al., 2015a) (Fig. 4, third vs fourth row), even though the latter relies on a high-performance deep neural network with features that are tuned to the statistics of natural images. Seen more broadly, this finding suggests that natural image generation does not necessarily depend on deep hierarchical representations or on the training of the feature maps. Instead, for texture synthesis, both aspects rather seem to serve as fine-tuning of the image representation.

One concern about the proposed single-layer multi-scale model is its computational inefficiency since it involves convolutions with spatially large filters (up to 55×55). A more efficient way to achieve receptive fields of similar size would be to use a hierarchical multi-layer net. We conducted extensive experiments with various hierarchical architectures and while the synthesis is indeed significantly faster, the quality of the synthesized textures does not improve compared to a single-layer model. Thus for a minimal model of natural textures, deep hierarchical representations are not necessary but they can improve the efficiency of the texture synthesis.

Our results clearly demonstrate that Gram matrices computed from the feature maps of convolutional neural networks generically lead to useful summary statistics for texture synthesis. The Gram matrix on the feature maps transforms the representations from the convolutional neural network into a stationary feature space that captures the pairwise correlations between different features. If the number of feature maps is large, then the local structures in the image are well preserved in the projected space and the overlaps of the convolutional filtering add additional constraints. At the same time, averaging out the spatial dimensions yields sufficient flexibility to generate entirely new textures that differ from the reference on a patch by patch level, but still share much of the small- and long-range statistics.

The success of shallow convolutional networks with random filters in reproducing the structure of the reference texture is remarkable and indicates that they can be useful for parametric texture synthesis. Besides reproducing the stationary correlation structure of the reference image ("perceptual similarity") another desideratum of a texture synthesis is to exhibit a large variety between different samples generated from the same given image ("variability"). Hence, synthesis algorithms need to balance perceptual similarity and variability. This balance is determined by a complex interplay between the choice of summary statistics and the optimization algorithm used. For example the stopping criterion of the optimization algorithm can be adjusted to trade perceptual similarity for larger variability.

In this paper we focused on maximizing perceptual similarity only, and it is worth pointing out that additional efforts will be necessary to find an optimal trade-off between perceptual similarity and variability. For the synthesis of textures from the random models considered here, the trade-off leans more towards perceptual similarity in comparison to Gatys et al. (2015a)(due to the simpler optimization) which also explains the superior performance on some samples. In fact, we found some anecdotal evidence (not shown) in deeper multi-layer random CNNs where the reference texture was exactly reconstructed during the synthesis. From a theoretical point of view this is likely a finite size effect which does not necessarily constitute a failure of the chosen summary statistics: for finite size images it is well possible that only the reference image can exactly reproduce all the summary statistics. Therefore, in practice, the Gram matrices are not treated as hard constraints but as soft constraints only. More generally, we do not expect a perceptual distance metric to assign exactly zero to a random pair of patches from the same texture. Instead, we expect it to assign small values for pairs from the same texture, and large values for patches from different textures. Therefore, the selection of constraints is not sufficient to characterize a texture synthesis model but only determines the exact minima of the objective function (which are sought for by the synthesis). If we additionally consider images with small but non-zero distance to the reference statistics, then the set of equivalent textures increases substantially, and the precise composition of this set becomes critically dependent on the perceptual distance metric.

Mathematically, parametric texture synthesis models are described as ergodic random fields that have maximum entropy subject to certain constraints Zhu et al. (1997); Bruna & Mallat (2013); Zhu et al. (2000) (MaxEnt framework). Practical texture synthesis algorithms, however, always deal with finite

size images. As discussed above, two finite-size patches from the same ergodic random field will almost never feature the exact same summary statistics. This additional uncertainty in estimating the constraints on finite length processes is not thoroughly accounted for by the MaxEnt framework (see discussion on its "ad hockeries" by Jaynes (Jaynes (1982))). Thus, a critical difference of practical implementations of texture synthesis algorithms from the conceptual MaxEnt texture modeling framework is that they genuinely allow a small mismatch in the constraints. Accordingly, specifying the summary statistics is not sufficient but a comprehensive definition of a texture synthesis model should specify:

1. A metric $d(\mathbf{x}, \mathbf{y})$ that determines the distance between any two arbitrary textures $\mathbf{x}, \mathbf{y}$.
2. A bipartition $P_{\mathbf{x}}$ of the image space that determines which images are considered perceptually equivalent to a reference texture $\mathbf{x}$. A simple example for such a partition is the ϵ-environment $U_\epsilon(\mathbf{y}) := \{\mathbf{y} : d(\mathbf{y}, \mathbf{x}) < \epsilon\}$ and its complement.

This definition is relevant for both under- as well as over-constrained models, but its importance becomes particularly obvious for the latter. According to the Minimax entropy principle for texture modeling suggested by Zhu et al Zhu et al. (1997), as many constraints as possible should be used to reduce the (Kullback-Leibler) divergence between the true texture model and its estimate. However, for finite spatial size, the synthetic samples become exactly equivalent to the reference texture (up to shifts) in the limit of sufficiently many independent constraints. In contrast, if we explicitly allow for a small mismatch between the summary statistics of the reference image and the synthesized textures, then the set of possible textures does not constitute a low-dimensional manifold but rather a small volume within the pixel space. Alternatively, instead of introducing an ϵ-environment it is also possible to extent the MaxEnt framework to allow for variability in the summary statistics (Joan Bruna, personal communication). It will be interesting to compare in the future to what extent the difference between the two approaches can lead to differences in the perceptual appearance of the textures.

Taken together we have shown that simple single-layer CNNs with random filters can serve as the basis for excellent texture synthesis models that outperform previous hand-crafted synthesis models and sometimes even rivals the current state-of-the-art. This finding repeals previous observations that suggested a critical role for the multi-layer representations in trained deep networks for natural texture generation. On the other hand, it is not enough to just use sufficiently many constraints as one would predict from the MaxEnt framework. Instead, for the design of good texture synthesis algorithms it will be crucial to find distance measures for which the ϵ-environment around the reference texture leads to perceptually satisfying results. In this way, building better texture synthesis models is inherently related to better quantitative models of human perception.

REFERENCES

M. Aittala, T. Aila, and J. Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics*, 35, 2016.

G. Berger and R. Memisevic. Incorporating long-range consistency in cnn-based texture generation. Jun 2016.

Joan Bruna and Stéphane Mallat. Audio texture synthesis with scattering moments. *CoRR*, abs/1311.0407, 2013. URL `http://dblp.uni-trier.de/db/journals/corr/corr1311.html#BrunaM13`.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *ArXiv e-prints*, June 2016.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, and other contributors. Lasagne: First release., August 2015. URL `http://dx.doi.org/10.5281/zenodo.27878`.

L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*, May 2015a. URL `http://arxiv.org/abs/1505.07376`.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. Aug 2015b. URL http://arxiv.org/abs/1508.06576.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*., 2010.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pp. 229–238, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218446. URL http://doi.acm.org/10.1145/218380.218446.

E.T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, Sept. 1982. ISSN 0018-9219.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed 2016-05-12].

B. Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, February 1962. ISSN 0096-1000. doi: 10.1109/TIT.1962.1057698.

G. Liu, Y. Gousseau, and G. Xia. Texture synthesis through convolutional neural networks and spectrum constraints. May 2016.

Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision*, 40(1):49–70, October 2000. ISSN 0920-5691. doi: 10.1023/A:1026553619983. URL http://dx.doi.org/10.1023/A:1026553619983.

Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit, 2009.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. *arXiv:1603.03417 [cs]*, March 2016. URL http://arxiv.org/abs/1603.03417. arXiv: 1603.03417.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. *ArXiv e-prints*, January 2016a.

A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *ArXiv e-prints*, June 2016b.

Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.

Song Chun Zhu, Xiuwen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient markov chain monte carlo-toward a 'trichromacy' theory of texture. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(6):554–569, 2000. doi: 10.1109/34.862195. URL http://dx.doi.org/10.1109/34.862195.

A APPENDIX

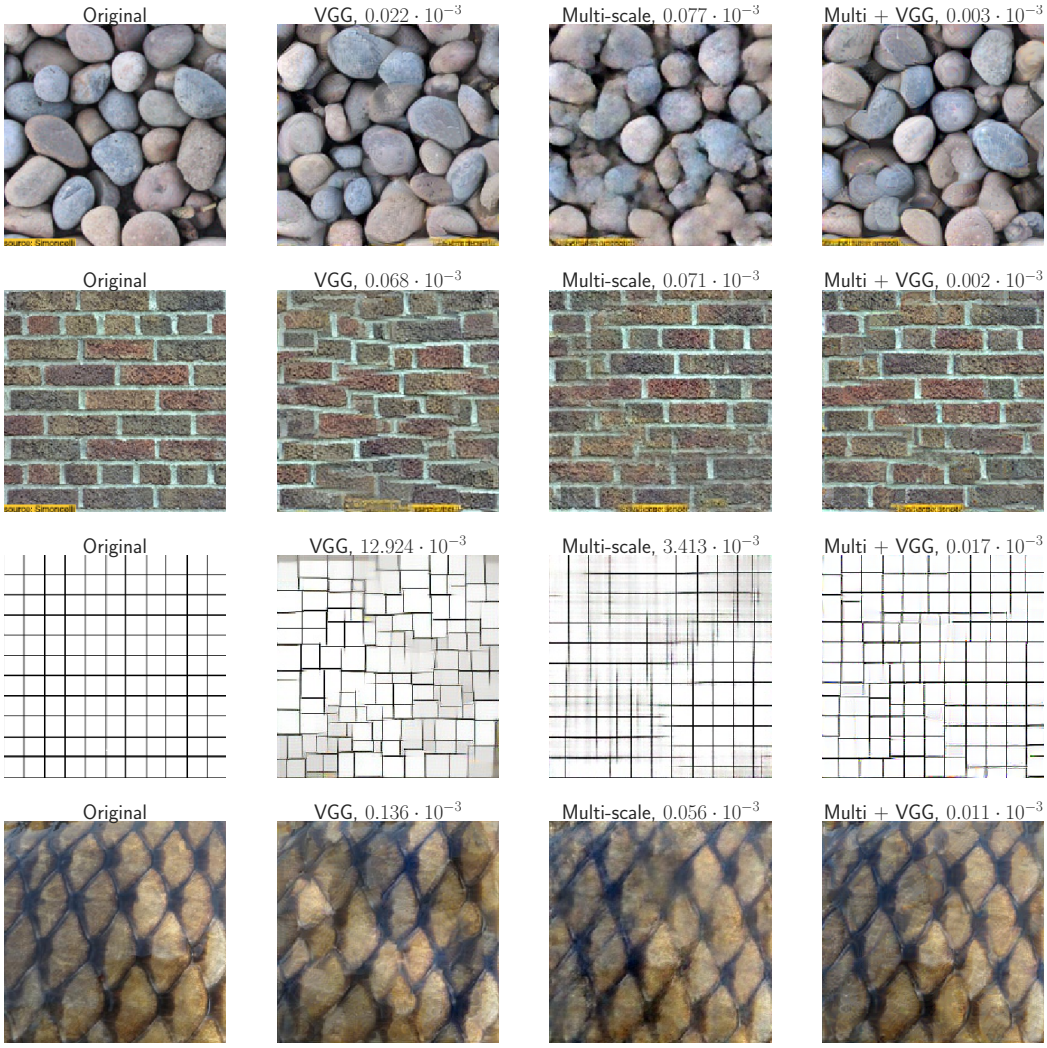

Figure 5: Initializing VGG-synthesis with a sample from the random multi-scale model. The first column shows the original textures, the second and third columns show samples from the standard VGG-based synthesis (random initialization) (Gatys et al., 2015a) and the random multi-scale model. The last column shows samples from the VGG-based model, which was initialized with samples from the random multi-scale model (from column 3). On top of all samples we report the corresponding values of the VGG-loss (7). Empirically, the VGG loss is up to two orders of magnitude lower in the last column relative to the standard VGG synthesis.

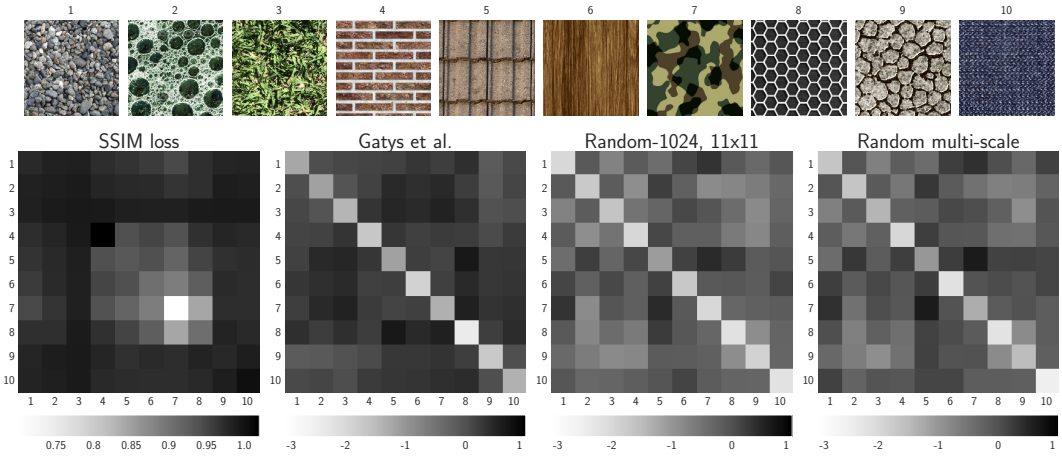

Figure 6: Similarity measures between textures computed using the structural similarity index on the pixels (SSIM, left column) and normalized euclidean distances in the feature spaces of VGG (second column) and two shallow texture models (third and fourth column). Ten random patches were extracted from each of ten different textures (examples of patches are shown in top row). The matrix element (i, j) of each similarity matrix corresponds to the median distance between patches from textures i and j. The values for all but the SSIM matrix are shown on a log-scale.

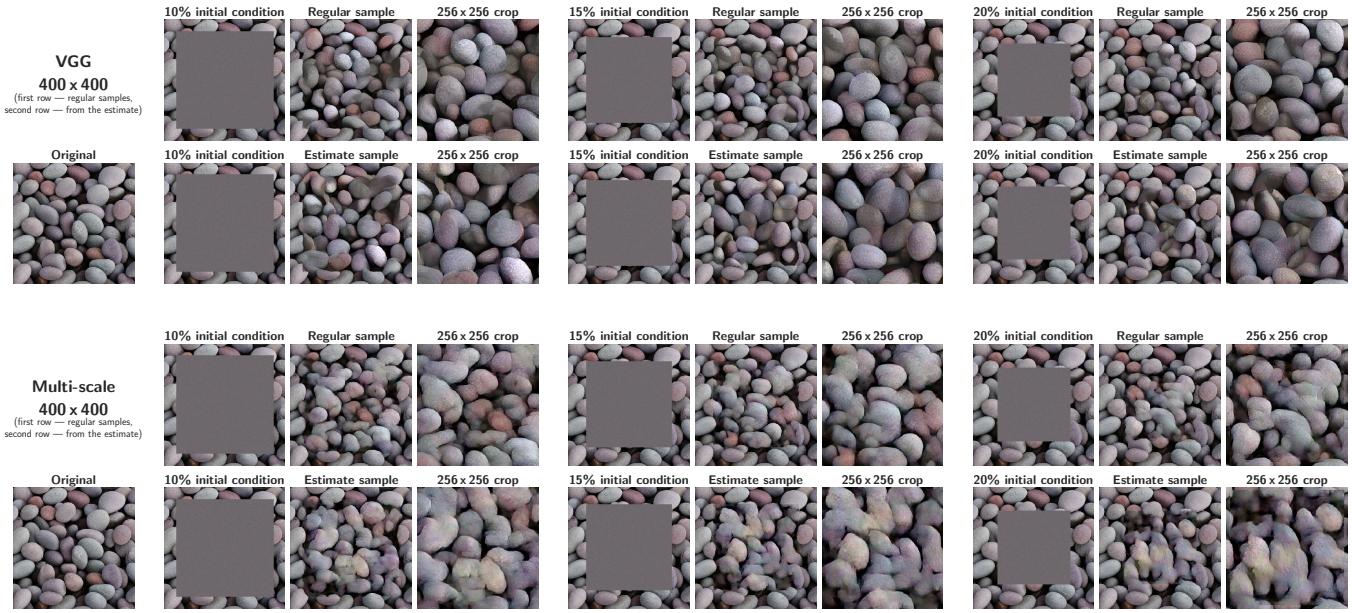

Figure 7: Examples of inpainted textures for the VGG model (Gatys et al., 2015a, , two top rows) and random multi-scale model (two bottom rows). Textures were inpainted starting from three different initial conditions (10 %, 15 %, 20 % corresponding to the width of the frame used for initialization), and for each initial condition the texture was inpainted either by matching the Gram matrix of the patch used for initializing the frame (regular sample) or by matching the Gram matrix estimated over many (500) randomly extracted patches from the texture (estimate sample).