GANosaic: Mosaic Creation with Generative Texture Manifolds

Nikolay Jetchev
nikolay.jetchev@zalando.de
Zalando Research

Urs Bergmann
urs.bergmann@zalando.de
Zalando Research

Calvin Seward
calvin.seward@zalando.de
Zalando Research

Abstract

This paper presents a novel framework for generating texture mosaics with convolutional neural networks. Our method is called GANosaic and performs optimization in the latent noise space of a generative texture model, which allows the transformation of a content image into a mosaic exhibiting the visual properties of the underlying texture manifold. To represent that manifold, we use a state-of-the-art generative adversarial method for texture synthesis [1], which can learn expressive texture representations from data and produce mosaic images with very high resolution. This fully convolutional model generates smooth (without any visible borders) mosaic images which morph and blend different textures locally. In addition, we develop a new type of differentiable statistical regularization appropriate for optimization over the prior noise space of the PSGAN model.

Mosaics are a classical art form. The Romans were masters in skillfully selecting small colored stones to make beautiful mosaics of large scenes. Later, in his paintings, the Renaissance painter Archimboldo composed various objects to make amazing portraits of people. In general, mosaics work because of the properties of the human visual system to average colors over spatial regions – when looking from a distance the large image emerges, but when looking closely the details of the single tiles emerge. In modern times, computer graphic algorithms have enabled different forms of digital image mosaics [12, 9]. However, these methods use distinct non-overlapping small tiles to paint the large image. A seamless mosaic style like Archimboldo's – where the whole image acts as a mosaic, without any tiles with borders – is visually closer to modern methods of texture synthesis and transfer.

Image quilting [4] recombines patches from the original textures in order to smoothly reconstruct a target image – "texture transfer". However, a disadvantage is the high runtime complexity when generating large images. In addition, since instance models merely copy the original pixels, they cannot be used to generalize and create novel textures from multiple examples.

The work of [5] uses discriminatively trained deep neural network as effective parametric image descriptors, allowing both texture synthesis and a novel form of texture transfer called "neural art style transfer." However, texture synthesis and transfer is performed from a single example image and lacks the ability to represent and morph textures defined by several different images.

Spatial versions of Generative Adversarial Networks (GAN) are well suited to unsupervised learning of textures [8, 1]. The Periodic Spatial GAN (PSGAN) allows high quality texture synthesis, with efficient memory and speed usage. It can also leverage information from many input images and use them to learn a texture manifold, a rich distribution over many textures allowing morphing into novel textures. Such generative models can give more widely varied outputs than the instance and neural descriptor based approaches to texture synthesis. Such variety is key to our proposed method.

Our novel GANosaic method has two steps. First, a PSGAN is trained on a set of example images, for details see Appendix I. Second, the generator G of the PSGAN is used as a module in an optimization problem: generate an image that is as close as possible to a "content" image I, while staying on the learned texture manifold, the "style". One way to do this is to adapt the input noise tensor Z s.t. the output image from the generator $G(Z)$ is as close as possible to the target image I. This is done by defining the distance between $G(Z)$ and I as a loss function and then optimizing it w.r.t. Z. However,

Figure 1: A texture manifold was learned with a PSGAN from satellite images of the city of Sydney, serving as texture prior for our neural mosaic. GANosaic renders a human portrait in a mosaic of size 1984x1472 pixels by minimizing a loss function – paint the target image and stay close to the texture manifold – w.r.t. the texture generator input tensor, with 62x46 spatial dimensions and 60 channels. The perceptual distance in the loss is represented by layer Conv5-1 of the pre-trained VGG [13] network. The mosaic is best seen zoomed-in to appreciate the rich small details of both city and forest.

during PSGAN training Z came from a prior noise distribution, and the mosaic optimization can push Z to values far away from the prior distribution and lead to degenerate looking textures.

For aesthetically pleasing mosaics we want $G(Z)$ to stay as close as possible to the texture manifold "style". When we use a PSGAN texture model, the mosaic style is better represented with a texture loss term that ensures that the input Z tensor[1] stays close to its prior statistics used during PSGAN training. Concretely, we model the loss to ensure that the local channels Z^l keep their statistical independence. The total loss function is therefore composed of two parts, a content loss and a texture loss:

$$\mathcal{L}(Z) = \mathcal{L}_c(Z) + \alpha_l \mathcal{L}_{tex}(Z^l) \tag{1}$$

The term $\mathcal{L}_c$ denotes the content loss ensuring reconstruction of I by $G(Z)$:

$$\mathcal{L}_c(Z) = \|\phi(I) - \phi(G(Z))\|_F^2 \tag{2}$$

Here $\|.\|_F^2$ denotes the mean of all squared tensor elements. The mapping ϕ is the "correspondence map" [4], which specifies what perceptual distance metric we want to use w.r.t. the content image. This can be a simple predefined image transformation, or a more complex approach, e.g. utilizing the outputs of pretrained convolutional filters [5]. By using some image downscaling operator in ϕ (e.g. pooling layers) we split the frequencies of the resultant image: the low frequencies are determined by the content image, and the high frequencies come from the texture manifold. Such a split improves the mosaic quality, see Appendix II for ablation studies regarding the effects of the choice of ϕ.

The texture loss is also required to keep the optimized noise tensor close to the manifold of textures created by the prior noise distribution. It regularizes the local noise channels Z^l of the Z noise tensor $Z = [Z^g, Z^l, Z^p]$. See Appendix III for more details on the loss $\mathcal{L}_{tex}$ and its effect on mosaic output.

In summary, the GANosaic is a powerful novel method to generate art, with the following key properties:

- generation of seamless mosaics with unique texture visual aesthetics
- flexible differentiable texture model [1] that learns and morphs diverse texture images
- very large scalability with respect to output mosaic size – all calls to the generator $G(Z)$ can be efficiently split into small tensor chunks seamlessly forming a very large final image [8]
- fast optimization in Z noise space with quick convergence

Please see Figure 1 for an example of mosaics that can be created by our method. Appendix IV has additional discussion of the properties of GANosaic.

---

[1]In our notation Z denotes both the noise random variable and a tensor sampled from it.

Acknowledgments

The authors would like to thank Roland Vollgraf for useful discussions and feedback.

References

[1] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial GAN. In *Proceedings of The 34th International Conference on Machine Learning*, 2017.

[2] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[3] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016.

[4] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, 2001.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, 2014.

[7] A. Hyvärinen, J. Hurri, and P. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.* Springer, 2009.

[8] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *CoRR*, abs/1611.08207, 2016.

[9] J. Kim and F. Pellacini. Jigsaw image mosaics. In *Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, 2002.

[10] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[11] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[12] Robert Silvers. *Photomosaics*. Henry Holt and Co., Inc., 1997.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Appendix I

The PSGAN method

This section contains a brief summary of the texture model PSGAN, for elaboration see [1]. Generative Adversarial Networks [6] learn a generator network $G(z)$ to distort a noise vector z, which is sampled from a standard distribution (e.g. uniform), such that the distorted probability distribution is close to the distribution observed through the training samples of the form $X \in \mathbb{R}^{H \times W \times 3}$. This is achieved with a game theoretic idea, by letting the generator network G play against an additional network, the discriminator D: the task of the discriminator is to classify a sample as being from the generator or from the training set, while the generator tries to be as good as possible in producing samples that get classified by the discriminator as real training data.

The extensions of PSGAN beyond the standard GAN framework are threefold. First, as in spatial GANs [8], the architecture is chosen to be a fully convolutional version of DCGAN [11] and the noise vector z is extended to a spatial tensor $Z \in \mathbb{R}^{L \times M \times d}$. Here, L and M are spatial dimensions, while d is the channel dimension. Hence, akin to DCGAN, the fractionally strided convolutions in PSGANs upsample the spatial dimensions L and M to the output dimensions H and W. In our case we typically use 5 convolution layers with each a fractional stride of $\frac{1}{2}$, hence the total upsampling in our case is $\frac{H}{L} = \frac{W}{M} = 2^5 = 32$.

As the discriminator is chosen symmetrically to G, in particular also fully convolutional, the standard GAN cost function needs to be marginalized over the spatial classifications:

$$V(D, G) = \frac{1}{LM} \sum_{\lambda=1}^{L} \sum_{\mu=1}^{M} \mathbb{E}_{Z \sim p_Z(Z)} \left[\log \left(1 - D_{\lambda\mu}(G(Z)) \right) \right]$$
$$+ \frac{1}{LM} \sum_{\lambda=1}^{L} \sum_{\mu=1}^{M} \mathbb{E}_{X' \sim p_{\text{data}}(X)} \left[\log D_{\lambda\mu}(X') \right], \quad (3)$$

where $D_{\lambda\mu}(X)$ is the discriminator output at location λ and μ. The key advantage of this approach is that the image patches used for the training minibatches can have different size than the image outputs used when sampling from the model, yielding arbitrary large output resolution. However, as the receptive field of a single location in the Z tensor is spatially limited in the output, far away regions are independently sampled. The local statistics must therefore be independent of the position – in other words, only sampling of a single texture is possible. To overcome this limitation, as a second extension, a fraction of channels in Z are spatially shared to allow for conditioning on a global structure. The final extension is to implement a spatial basis in parts of Z, which can be used to anchor image generation. It has been shown in [1] that a plane wave parameterization for the spatial basis allows the generation of periodic textures, and can also lead to better quality of non-periodic textures. The wave numbers of the plane waves in PSGAN are given as functions of the global channels by a multi-layered perceptron, which is learned end-to-end alongside G and D. In total the tensor $Z = [Z^g, Z^l, Z^p]$ consists of three parts: a local part Z^l, a global part Z^g, and a periodic part Z^p, concatenated in the channel dimension.

After learning, the spatially shared global channels define a texture to be sampled, while the independent samples in the local dimensions give rise to local pattern variation. Importantly, when the global channels are allowed to change smoothly in the spatial dimensions, this yields a spatial transitioning of textures, while being locally still plausible textures. Hence we speak of a texture manifold.

Examples of textures learned by PSGAN

Figure 3 shows how the texture manifolds learned by PSGAN look. We use input texture images for PSGAN training of Sydney satellite images from Google Maps, stone wall images from Wikimedia Commons, and DTD "scaly" from [2].

Technical note: fixed constant batch normalization in G for GANosaic

The generator G of PSGAN was trained using batch normalization after every deconvolutional layer. The batch normalization calculates per layer statistics that capture the distribution of feature activations in the minibatch input to G. However, the GANosaic method (optimization of Z w.r.t. a content and texture loss, Z is a tensor with 1 instance of a large spatial extent) is different than the PSGAN training
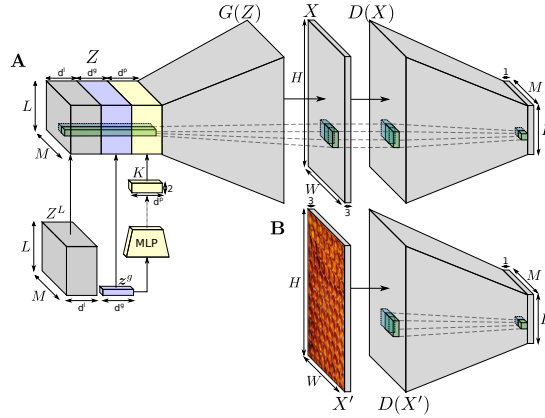
Figure 2: Illustration of the PSGAN model. **A** The fully convolutional generator network $G(Z)$ maps a spatial tensor $Z_{\lambda\mu i}$, λ and μ being the spatial indices, to an input image X. Every subvector at a spatial location in Z, e.g. the blue or green columns in the Figure, map to a limited area in X. As usual in GAN training, the discriminator gets either a generated image X or, as in **B**, an image patch X' from the real data.

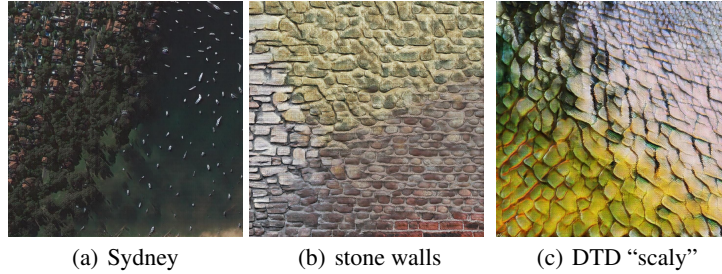| (a) Sydney | (b) stone walls | (c) DTD "scaly" |

Figure 3: Examples of the PSGAN texture models used for our mosaics. Morph plots show the ability of texture manifolds to smoothly change texture processes. The plots (size 960x960 pixels) created by bi-linearly interpolating the Z^g tensor (size $L = M = 30$) between 4 random texture samples.

(optimization of G parameters w.r.t. GAN loss function, Z has many instances of spatially small arrays from the noise prior distribution). We found empirically that GANosaic works better by using fixed statistics (mean and variance) for the batch normalization operations of the trained G. Concretely, we pre-calculate the batch normalization statistics of a batch with many instances of Z sampled from the prior. Afterwards, these statistics are used as a constant rescaling for each batch normalization operation inside the network G.

Appendix II

Selecting the perceptual distance metric

The correspondence map ϕ used for the content loss encodes directly a choice of perceptual distance metric – it allows us to have flexibility in the transfer of the texture appearance on the mosaic. If we use the identity as map, then for some images the mosaic output will be degenerate. Figure 4(a) shows this drawback of the identity correspondence map when the content image has too high frequencies, which are difficult to map to the texture manifold. Adding downscaling (e.g. with an average pooling filter) to ϕ will emphasize the lower frequencies in the content image and lead to better texture appearance, see Figure4(b,c). Downscaling too much will make the mosaic reproduce the content image less accurately as in Figure 4(d), but as a trade-off the stone textures are really well recognizable.

Another interesting choice for ϕ may be to use instead of the exact colors the luminance of the RGB image, defined by us as the average per pixel of the 3 color channels. This is useful when the texture

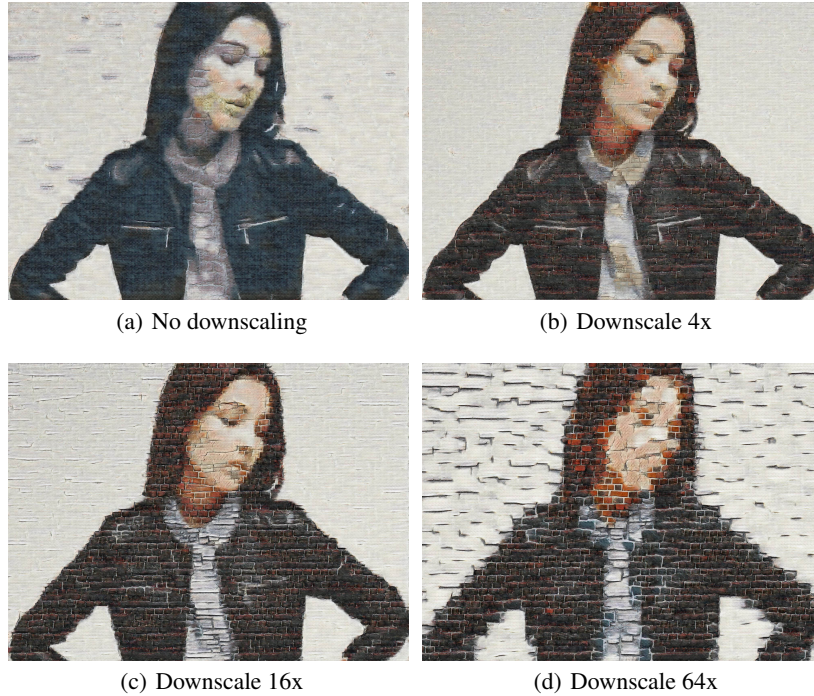| (a) No downscaling | (b) Downscale 4x |
| (c) Downscale 16x | (d) Downscale 64x |

Figure 4: Downscaling by average pooling in the correspondence map ϕ allows a tradeoff in the content loss: accurate content image rendering (a) or strong texture aesthetics (d). Mosaics (b) and (c) are more balanced. All 4 mosaics have the resolution 1984x1472 pixels and use 7 images of stones from Wikimedia Commons for textures.

manifold is very different in color hue from the content image. Figure 5(a) shows an example mosaic with the luminance map, and it has more color variation than the RGB map 5(d).

The downscaling and color transformations are examples of manually specified correspondence maps ϕ. As an alternative, we can take filters from the pretrained VGG network [13], which is similar to the approach of [5]. Figure 5(b,e,f) shows our results with VGG, which has a different aesthetics than the other choices of ϕ – this perceptual distance is more flexible w.r.t. color hue and also is more flexible w.r.t. spatial matching of the content. This is due to the VGG network architecture, which is deep (many convolutional layers) and wide (many filter channels) and uses pooling operations.

Disabling the mosaic content loss

A very specific choice of correspondence map will use a map ϕ which always outputs a constant value, equivalent to disabling the whole content loss term in the GANosaic loss. In that case, we introduce a simpler alternative method that can create mosaic images using a PSGAN texture generator. We can directly paint the noise global dimensions Z^g with values related to the pixels of the content image. E.g. we can use a random linear projection from pixels (3 channels) to the d^g channels of the global noise tensor, followed by a nonlinearity to keep the values in $[-1, 1]$. Concretely, given an image I of size $H \times W$ pixels, we can calculate a downscaled version $\hat{I}$ of size $L \times M$ pixels – the spatial resolution of the latent noise space of G. We can then sample a random matrix $R \in \mathbb{R}^{d^g \times 3}$ and calculate per spatial position $Z^g_{\lambda,\mu} = \sin(R\hat{I}_{\lambda,\mu})$. Afterwards we can generate an image $G(Z)$ with the projection Z^g and local noise Z^l from the prior.

While very simple, this approach is useful for exploration of texture manifolds and is very fast to calculate. Figure 6 shows an example: the random projection paints the low frequency segments of the content image, while exhibiting a lot of details from the texture manifold. This is useful for creating smoothly morphing videos illustrating random walks in the space of the texture manifolds, while projecting a specific content image, and usually preserving well the low frequencies of the content. At `youtu.be/4GAFQwE3kLs` you can see an example video of Che Guevara rendered with Sydney

(a) Luminance, Downscale 4x   (b) Conv4-2

(c) RGB, Downscale 16x   (d) RGB, Down. 4x
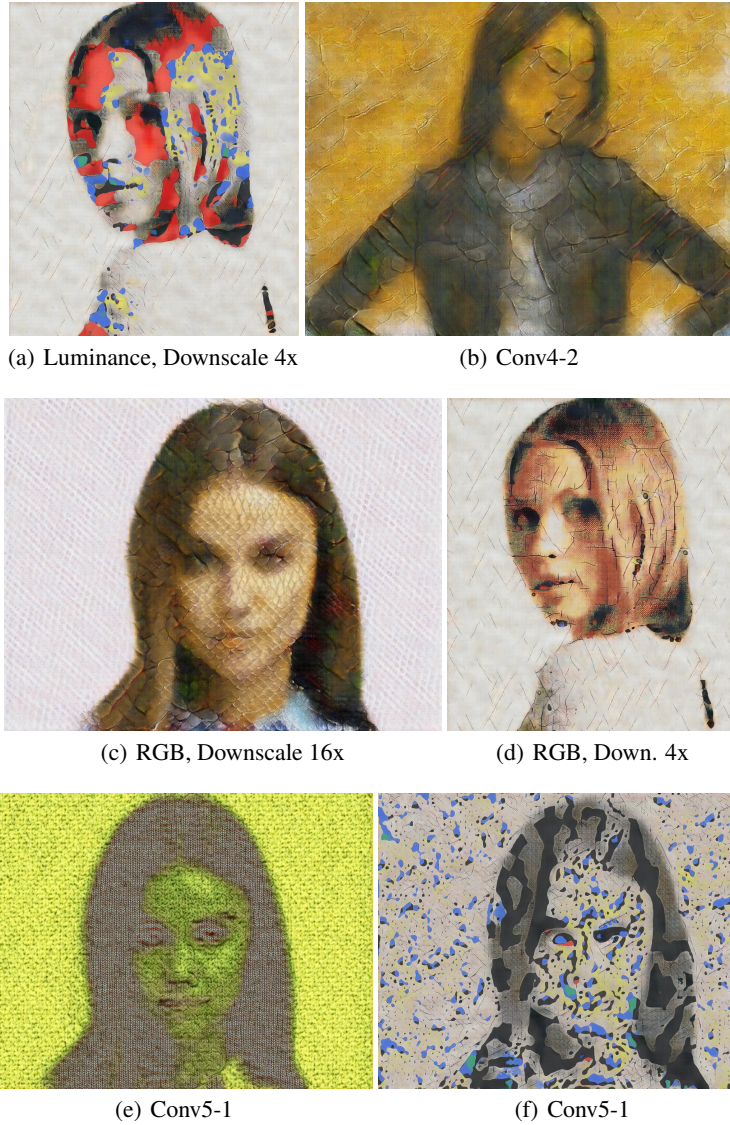
(e) Conv5-1   (f) Conv5-1

Figure 5: Results using different correspondence maps. (a) shows how using only the luminance allows to make mosaics with a more experimental look, away from the original color palette as used in (d). In (b,e,f) layers of the VGG network encode more complex perceptual distances. Used textures: DTD "knitted" in e), DTD "scaly" in b,c), Juan Miro paintings in a,d,f). Please zoom-in for details.

Figure 6: Random projections are useful for texture exploration, here the Sydney satellite views. A content image of size 64x48 was projected to global noise channels Z^g of the same size spatially, and local noise Z^l from the prior. The resulting $G(Z)$ generates a 2048x1536 pixel output image.

7

satellite textures. As a drawback, the random projection mosaics completely ignore the high frequencies of the content image. The random projection output is also more unpredictable: some projections look better than others, but the fast generation speed (0.1s for the 2048x1536 pixel image in Figure 6) enables exploring many such images as artistic selection process.

Appendix III

Texture loss: keeping the input tensor Z^l close to the prior noise distribution

Optimization of Z^l w.r.t. the content loss $\mathcal{L}_c$ can introduce spatial correlations between the local dimensions. During training of the texture model, however, the local dimensions Z^l in the PSGAN model were independently sampled from the prior at every spatial position and channel dimension. Hence, the correlations imply a move away from the learned texture manifold. To remedy this, we introduce a regularization term: the key idea is that samples taken from the joint distribution of neighboring local dimensions should be distributed according to the prior distribution during training (up to finite sample size effects). In our case, as the prior is an independent uniform distribution, this means the samples should fill up the whole hypercube $[0, 1]^{L \times M}$. In contrast, if local dimensions were perfectly correlated, the samples would lie exclusively on the diagonal of the hypercube.

To implement this idea we assume independence in the channel dimension and considered the different channels as the samples. By employing a kernel density estimate, the joint distribution can be estimated and compared to the prior distribution. For practical reasons, only pairwise neighboring positions in Z^l are evaluated. The restriction to neighboring positions can be justified by noting that correlations in natural images fall off monotonously with distance [7]. The computational benefit is a reduction of quadratic to linear time complexity. Formally, we can write the texture loss term $\mathcal{L}_{tex}$ as:

$$\mathcal{L}_{tex}(Z^l) = \sum_{\lambda=1,\mu=1}^{L-1,M-1} \sum_{\Delta\lambda,\Delta\mu \in \Delta} d\left(\hat{p}_{\left[Z^l_{\lambda,\mu}, Z^l_{\lambda+\Delta\lambda,\mu+\Delta\mu}\right]}, \hat{p}_{\text{prior}}\right) \tag{4}$$

where $d(p_1, p_2)$ measures the distance of two probability distributions p_1 and p_2, and the square brackets denote the concatenation of column vectors to a matrix. The set of spatial offsets Δ determines for which neighboring positions the distribution is regularized. We took $\Delta = \{(0, 1), (1, 1), (1, 0)\}$.

The kernel density estimate given $\hat{Z} \in \mathbb{R}^{d_l \times 2}$ and evaluated at a point $\boldsymbol{\tau}$ is given as $\hat{p}_{\hat{Z}}(\boldsymbol{\tau}) = \frac{1}{d_l \sigma} \sum_{i=1}^{d_l} k\left(\frac{\|[\hat{Z}_{i1}, \hat{Z}_{i2}] - \boldsymbol{\tau}\|^2}{\sigma}\right)$. Any valid kernel function can be used for k; we employed a Gaussian kernel. From the form of $\hat{p}_{\hat{Z}}$ we see that the target probability distribution of the regularizer is the convolution of the prior probability distribution with the Gaussian kernel, i.e. $\hat{p}_{\text{prior}} = p_{\text{prior}} * k$.

Finally, the distance function d between the probability distributions needs to be defined. We simply calculate the distance as the square difference between the two distributions evaluated on the set $\mathcal{G}$ of grid points equally spaced in the unit cube, i.e. $d(p_1, p_2) = \sum_{\boldsymbol{\tau} \in \mathcal{G}} \|p_1(\boldsymbol{\tau}) - p_2(\boldsymbol{\tau})\|^2$. This makes the regularizer a differentiable function w.r.t. Z. Figure 7 gives a toy example of the behavior of the regularizer. Note that the regularizer is similar to determinantal point processes [10], in particular the resulting samples tend to be too regular in comparison to samples from the prior.

Visualizing the effects of local dimensions optimization and texture loss

Optimizing the local values Z^l together with the global Z^g can lead to lower content loss than tuning Z^g alone and fixing Z^l to a tensor sampled from the prior. However, as discussed in the first section in Appendix III, special care needs to be taken to keep the distribution of the Z^l close to the prior distribution and avoid degeneracy from the texture manifold – thus the texture loss term $\mathcal{L}_{tex}$. Figure 8 shows the mosaic quality obtained by optimizing and regularizing Z^l. The regularization is important in preventing degeneration.

Appendix IV

This section contains a short discussion of the properties of GANosaic mosaics. Traditionally, photomosaic algorithms utilized large image datasets [12, 9]. Texture rendering is usually done with a single texture [4, 5]. In contrast, GANosaic can use rich texture manifolds as style representations
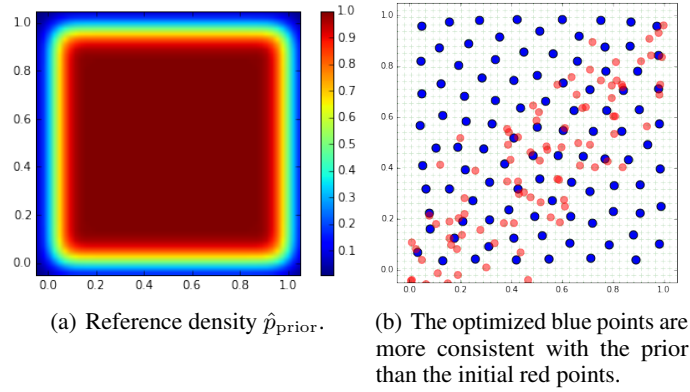
(a) Reference density $\hat{p}_{\mathrm{prior}}$.

(b) The optimized blue points are more consistent with the prior than the initial red points.

Figure 7: (a) The reference density $\hat{p}_{\mathrm{prior}} = p_{\mathrm{prior}} * k$, using the uniform distribution on $[0,1] \times [0,1]$ as prior p_{prior}, and a Gauss kernel k with $\sigma = 0.1$ for density estimation. (b) 100 samples (red dots) are drawn from a different distribution than the prior. We use a grid with $N = 40 \times 40$ regularly spaced points (green crosses) to measure the distance of the reference density and the kernel density estimate of the samples. Minimizing the distance yields samples (blue dots), which are more consistent with the prior p_{prior}.

(a) No Z^l optimisation.

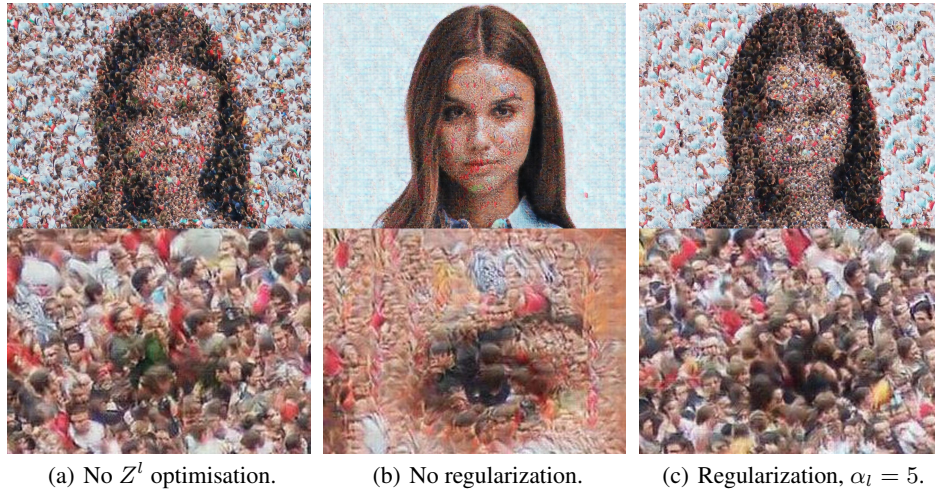(b) No regularization.

(c) Regularization, $\alpha_l = 5$.

Figure 8: Different mosaic results, with a zoom-in on the right eye for details. (a) optimizes only the global dimensions and has worse content fit than (b) and (c). In (b) optimizing the local dimensions fits the content better, but degeneration occurs: see how the people in the texture become blurry. In (c) regularization corrects the degeneration, while still fitting the content better than (a). A texture of many small people in a crowd is used, which emphasizes the visual degeneracy (b) or correct random-like texture appearance (a,c).

and allows the generation via convolutional neural networks of large mosaics smoothly rendering any content image. The generation of texture mosaics is achieved by optimization in the latent noise space of a PSGAN texture model. An application of the GANosaic can be consumer-facing apps that allow quick rendering of user content (images and video), using pre-trained texture models. Another more professional use case can be graphical design: the artist can use PSGAN and GANosaic as tools in the creative process:

- select carefully a set of texture examples
- train a PSGAN texture model on them
- use the texture model on any content image to create high resolution art, suitable for posters.

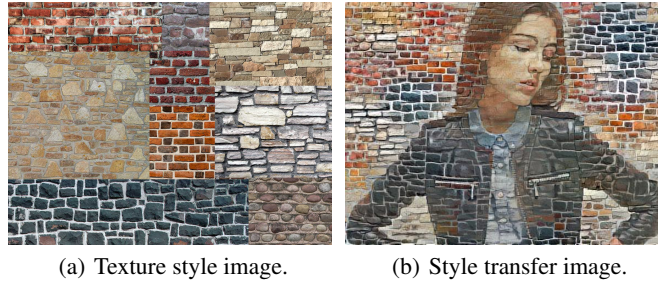(a) Texture style image.   (b) Style transfer image.

Figure 9: Using the (composite) style image (a), and a human photography as content (original shown in Figure 1) the neural style transfer method produces the image shown in (b). A detailed look at that output image reveals interesting artifacts. The jacket segment is a good stone mosaic, but the face segment has issues: it is rendered too similar to the content pixels and lacks the stone texture.

The fast runtime and the ability to handle arbitrarily large output image resolutions comes from the spatial GAN architecture [8] used in the texture prior of GANosaic. The generation of $G(Z)$ can be easily split by calculating G on separate subtensors of Z and combining the results. This property applies also to our optimization framework, since all the loss function terms can be calculated and aggregated in spatial chunks. Thus, mosaics of very high resolution, practically unlimited by memory (only by storage) can be made. The computation time scales linearly with the number of output pixels.

In order to minimize the mosaic loss in Equation 1, different spatial regions of the input noise tensor Z^g will converge to different values and thus the image output from the generator will be a mosaic of different texture processes. While this works well for the low frequencies in the content image, there is a limit to how high the frequencies in the output mosaic can be. The receptive field of the generator model determines the highest possible frequencies we can obtain in the mosaics by setting the generator input. Optimizing both the local and global dimensions improves the texture mosaic resolution and allows to paint finer details better fitting the content, but there is always some limit how high the mosaic image frequencies can go and what level of pixel detail is achievable.

Downscaling in the map ϕ can improve the image quality of the output mosaic (see Figure 4) – it acts like an averaging filter that removes the high frequencies from the content image. Thus, the content loss depends on the lower frequencies, and the high frequencies will be determined by the texture manifold. The choice of content image and its pixel size matter as well. Large content images have lower frequencies and are easier for mosaics. In practice, we can always upscale the input image to obtain larger mosaics with lower frequencies. Using a smaller texture brush relative to the image leads to a lower content loss, analogous to a large photomosaic with small tiles. On the other hand, smaller content images imply larger textures relative to them (e.g. the works of Archimboldo) and this is a more challenging case for mosaic optimization.

The approach of GANosaic to locally fit the "right" texture is different than the neural art style approach [5]. Figure 9 shows an example of neural style transfer when using a style image with many textures. The style descriptor consists of the feature correlations marginalized over the spatial extent of the image, and style transfer will try to transfer the full distribution of the style image to the content – thus the mixed stone background. The pixel space optimization can preserve the high frequency details of the content image, but in some regions (e.g. the face in Figure 9 b)) it looks like merely painting directly the content, rather than trying to represent it with textures.

In contrast, GANosaic works by modifying *only* the input noise tensor Z to the generator network G. This network acts as a regularizer that outputs images $G(Z)$ that are close to the underlying PSGAN texture manifold. GANosaic does not optimize directly the pixels of the output mosaic image, but only modifies Z as input of G. This is a more constrained optimization problem since the noise space Z has much lower spatial size than the image I. E.g. with G with 5 layers an image with 1024x1024 pixels will be generated by a tensor Z of size 32x32 spatial positions.

In a related work, [3] explored the blending of multiple styles by parametrically mixing their statistical descriptors, but the result is a style averaging globally the properties of the input styles, rather than painting locally with different styles. In contrast, the GANosaic (e.g. as shown in Figure 4 with the same texture style) can focus on different modes from the texture distribution that well approximates the content image locally, and the regularization ensures that each local region is on the texture manifold.