Fourier-CPPNs for Image Synthesis

Mattie Tesfaldet, Xavier Snelgrove, David Vazquez
Element AI
Montréal, Canada
mattie,xavier.snelgrove,dvazquez@elementai.com

Abstract

Compositional Pattern Producing Networks (CPPNs) are differentiable networks that independently map (x,y) pixel coordinates to (r,g,b) colour values. Recently, CPPNs have been used for creating interesting imagery for creative purposes, e.g. *neural art. However their architecture biases generated images to be overly smooth, lacking high-frequency detail. In this work, we extend CPPNs to explicitly model the frequency information for each pixel output, capturing frequencies beyond the DC component. We show that our Fourier-CPPNs (F-CPPNs) provide improved visual detail for image synthesis.*

1. Introduction

The fields of computer graphics and computer vision have a long history of introducing new computational approaches for the creation of images. Recently, exciting new deep learning approaches for synthesizing images have come about, such as Generative Adversarial Networks (GANs) [5] for generating realistic faces [9, 10], Convolutional Networks (ConvNets) for image style transfer [4, 16, 7], and Compositional Pattern Producing Networks (CPPNs) [15, 8, 6, 11, 12, 14] for creating aesthetically interesting high-resolution images for creative purposes. The proposed research extends CPPNs by explicitly modelling frequency information. Our experiments show that our Fourier-CPPN (F-CPPN) produces images with improved visual detail while maintaining the advantages of CPPNs.

1.1. Compositional Pattern Producing Networks (CPPNs)

CPPNs are differentiable networks that independently map (x,y) pixel coordinates to (r,g,b) colour values via the composition and combination of various simple activation functions (Fig. 1 top), *e.g.* linear, exponential, periodic, etc. Originally, CPPNs were designed for analysing the properties of natural developmental encodings [15] and

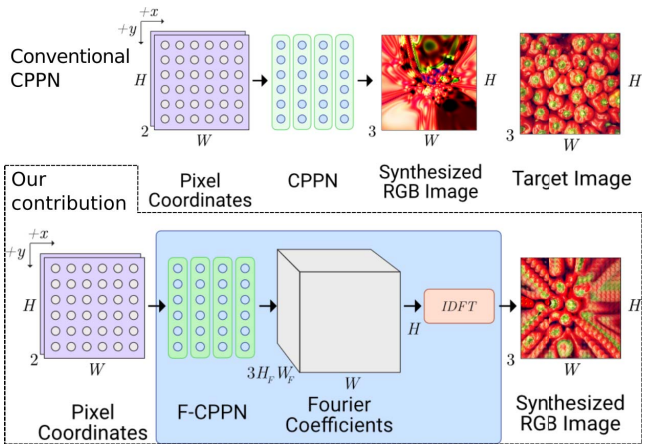

Figure 1: Fourier-CPPNs (F-CPPNs) for image synthesis. (top-left) CPPNs are differentiable networks that map (x,y) pixel coordinates to (r,g,b) colour values via linear and non-linear transformations. (bottom) We propose F-CPPNs, an extension of CPPNs which explicitly model the frequency information for each pixel output, capturing frequencies beyond what can be captured by CPPNs. This allows for outputs with increased visual detail.

were optimized using a neural-evolutionary approach that augmented the CPPN's weights *and* topology. Instead of using the same activation function at each layer, the evolutionary process would find the optimal activation from a list of possible activation functions and "grow" the network, increasing its complexity. Recently however, CPPNs have started to be used for creating interesting imagery for creative purposes [6, 11, 12, 14], *e.g.*, neural art. These recent CPPN implementations have avoided using neural-evolutionary approaches in determining the network's architecture and have opted for using a fixed architecture instead, whose weights are optimized via gradient descent.

CPPNs have several useful properties. The image they parameterize can be generated at arbitrary resolutions and at arbitrary crops. They can also easily integrate into computer graphics pipelines [14]. However, the locality of pixel coordinates and choice of smooth activation functions constrains

the resulting image such that colour tends to smoothly vary across neighbouring pixels. As a consequence of this inductive bias, CPPNs create images that appear overly smooth, lacking high-frequency detail that can otherwise add realism to the synthesized image. Up until now, CPPNs have not been designed to incorporate frequency information beyond the DC (zero-frequency) component. We propose an extension to CPPNs, based on Fourier analysis, where each pixel's colour value is represented as a linear combination of complex-valued sinusoids.

1.2. Fourier synthesis

Here we briefly review the relevant theory and mathematics before introducing F-CPPNs. The two-dimensional (2D) discrete Fourier transform (DFT) allows us to represent an image, I, as a linear combination of complex-valued sinusoidal basis images with varying frequency. The mixing coefficients of these images are given by the complex-valued $F[\omega_x, \omega_y]$, which represent the magnitude and phase of the sinusoid with spatial frequency ω_x and ω_y. The inverse 2D DFT (IDFT) allows us to synthesize I from its Fourier coefficients, F. It is defined, per colour channel, as follows,

$$I_c(x,y) = \frac{1}{\sqrt{WH}} \sum_{\omega_x=0}^{W-1} \sum_{\omega_y=0}^{H-1} F_c[\omega_x, \omega_y] e^{i2\pi(\omega_x x/W + \omega_y y/H)} , \quad (1)$$

where $I_c(x,y)$ is the intensity at pixel (x,y) for a particular colour channel c, $F_c[\omega_x, \omega_y]$ is the Fourier coefficient for the given spatial frequencies ω_x and ω_y, and W and H are the width and height of the image in pixels, respectively. Note that the number of frequencies (and coefficients) corresponds to the number of pixels in the input image. F and I above are complex-valued functions. Since we seek to generate a real-valued image, in this work we simply take the real part of I.

2. Fourier CPPNs

We propose the *Fourier*-CPPN (F-CPPN), an alternate parameterization to CPPNs where each pixel's (r,g,b) colour value is obtained from an IDFT on a set of learned Fourier coefficients (see Fig. 1 bottom),

$$(x,y) \xrightarrow{F-CPPN} \big(F_r[\omega_x, \omega_y], F_g[\omega_x, \omega_y], F_b[\omega_x, \omega_y]\big] \xrightarrow{IDFT} (r,g,b) . \quad (2)$$

Consider a $H \times W \times 2$ grid of pixel coordinates as inputs to a F-CPPN. Recall that a $H \times W$ image can be defined by a single set of Fourier coefficients of the same dimensions. We design our F-CPPN to output a smaller number of coefficients with dimensions $H_F \times W_F$. However, we allow them to vary at each pixel coordinate. This localized and spatially-varying Fourier parameterization of the image is defined as follows,

$$I_c(x,y) = \frac{1}{\sqrt{W_F H_F}} \sum_{\omega_x=0}^{W_F-1} \sum_{\omega_y=0}^{H_F-1} F_{xyc}[\omega_x, \omega_y] e^{i2\pi(\omega_x x/W_F + \omega_y y/H_F)} , \quad (3)$$

where $F_{xyc}[\omega_x, \omega_y]$ represents the localized frequency information for spatial frequencies ω_x and ω_y at pixel coordinate (x,y) for colour channel c. The final (r,g,b) colour value is constructed as $I(x,y) = (I_r(x,y), I_g(x,y), I_b(x,y))$, from which the final $H \times W \times 3$ image I is obtained.

This image representation is overparameterized. Instead of representing the image with $H \times W$ Fourier coefficients, it is now represented with $H \times W \times H_F \times W_F$ localized Fourier coefficients. However, this parameterization has some useful properties. First, a region of an image containing a periodic texture with period W_F and H_F in x and y, respectively, can be represented with a constant set of localized Fourier coefficients at every pixel location in that region. Second, a region of transition from one periodic texture to another can be represented as an interpolation from one set of localized Fourier coefficients to another.

By combining the inductive bias of a CPPN, which tends towards constant or smoothly varying outputs, with the property of the localized IDFT, whereby regions with constant coefficients become regions of constant periodic texture, we arrive at our contribution, the F-CPPN. Our F-CPPN is able to explicitly model frequency information beyond the DC component. We can consider CPPNs as a special case of F-CPPNs, where $W_F = W_H = 1$, in other words, a F-CPPN that only captures the DC component.

Architecture design Our F-CPPN architecture builds on the CPPN implementation from Mordvintsev et al. [12]. Their network consists of eight 1×1 convolutional layers, each with 24 filters and each followed by an activation function $\phi(a) = \big(\arctan(a)/0.67,\ \arctan^2(a)/0.67\big)$, where a is the output from a convolution and $(\cdot,\ \cdot)$ is a channel-wise concatenation. Note that CPPNs can be implemented as ConvNets strictly using 1×1 convolutions. This implies that no information is shared between neighbouring pixels. Our F-CPPN differs from their CPPN in that the final layer does not directly output (r,g,b) colour values but instead outputs $\big(F_{xyr}[\omega_x, \omega_y], F_{xyg}[\omega_x, \omega_y], F_{xyb}[\omega_x, \omega_y]\big)$ Fourier coefficients, which are then fed to an IDFT to produce (r,g,b) colour values.

3. Experiments

Our goal is to improve the visual detail of images synthesized by CPPNs by explicitly modelling frequency information, arriving at F-CPPNs. We qualitatively evaluate the

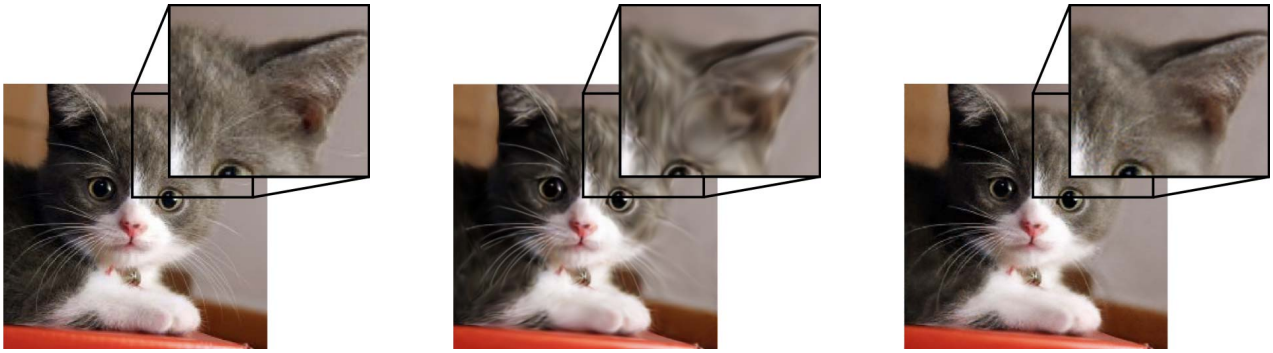

Figure 2: Image reconstruction via Compositional Pattern Producing Networks (CPPNs) and Fourier-CPPNs (F-CPPNs). (left) Target image. (middle) Mordvintsev et al. [12]'s CPPN output. (right) Our F-CPPN's output. Notice in the zoomed-in sections that our F-CPPN is able to better reconstruct the textural detail of the cat's fur.

F-CPPN approach of extending a CPPN's frequency representation beyond the DC component through an ablation study on two image synthesis tasks, image reconstruction and texture synthesis. For each task, we compare the outputs from our F-CPPN and Mordvintsev et al. [12]'s CPPN (the baseline). Additional results can be found in the supplementary material.

Training The objectives used for optimizing the weights of the F-CPPN and the baseline are described in the following sections. We used L-BFGS [1] for optimization. Results were generated using an NIVIDA Tesla P100 GPU and optimization took about an hour for generating a 224×224 image. The input pixel coordinates were set to range between $[\sqrt{3}, -\sqrt{3}]$ with $(0,0)$ in the centre. The weights for each layer were initialized randomly with zero mean and a variance equal to $\sqrt{1/C}$, where C is equal to the number of input activations. Biases were initialized to zero. The number of spatial frequencies, $H_F \times W_F$, was set to 10×10.

3.1. Image reconstruction

We show that F-CPPNs have the capacity to synthesize images with greater detail than the baseline with the straightforward task of image reconstruction. Both a F-CPPN and a CPPN are tasked with reconstructing a given image, I, to produce an output, $\hat{I}$. To optimize the weights of both networks, we use the content loss from Gatys et al. [4],

$$\mathscr{L}_{content} = \frac{1}{L} \sum_{l} ||\phi_l(I) - \phi_l(\hat{I})||_2^2 \,, \tag{4}$$

where $\phi_l(\cdot)$ are the activations of the l-th layer of VGG-19 [13] when processing input $(\cdot)$, L is number of layers used, and $||\cdot||_2$ is the L2 norm. In short, the content loss is computed as the mean squared error (MSE) between feature representations of I and $\hat{I}$. The activations used were from layers *conv1_1*, *pool1*, *pool2*, *pool3*, and *pool4* of VGG-19. As shown in Fig. 2, the level of detail captured by the F-CPPN is generally greater than the CPPN's.

3.2. Texture synthesis

A visual texture can be loosely defined as a region of an image with stationary feature statistics. Examples of natural textures can include bark, granite, or sand. Texture synthesis is the process of algorithmically generating new image regions that match the stationary feature statistics of a given source texture. Gatys et al. [3] demonstrated impressive results using the learned filters from VGG-19. Textures were modelled in terms of the normalized correlations between activation maps within several layers of the network. Here we synthesize textures with our F-CPPN and the baseline CPPN using Gatys et al. [3]'s texture objective. The task is as follows. Given a target texture, let $\mathbf{A}^l \in \mathbb{R}^{N_l \times M_l}$ be its row-vectorized activation maps at the l-th layer of a ConvNet (in this case, VGG-19). N_l and M_l denote the number of activation maps and the number of spatial locations, respectively. The normalized correlations between activation maps within a layer are encapsulated by a Gram matrix, $\mathbf{G}^l \in \mathbb{R}^{N_l \times N_l}$, whose entries are given by, $G^l_{ij} = \frac{1}{N_l M_l} \sum_{k=1}^{M_l} A^l_{ik} A^l_{jk}$. A^l_{ik} denotes the activation of feature i at location k in layer l on the target texture. Similarly, given a synthesized texture, let $\hat{\mathbf{A}}^l \in \mathbb{R}^{N_l \times M_l}$ be its row-vectorized activation maps and $\hat{\mathbf{G}}^l \in \mathbb{R}^{N_l \times N_l}$ be its Gram matrix, whose entries are given by, $\hat{G}^l_{ij} = \frac{1}{N_l M_l} \sum_{k=1}^{M_l} \hat{A}^l_{ik} \hat{A}^l_{jk}$. The final objective is defined as the average of the MSE between the Gram matrices of the target texture and that of the synthesized texture,

$$\mathscr{L}_{style} = \frac{1}{L} \sum_{l} \|\mathbf{G}^l - \hat{\mathbf{G}}^l\|_F^2 \,, \tag{5}$$

where L is the number of ConvNet layers used when computing Gram matrices and $\|\cdot\|_F$ is the Frobenius norm. This texture objective is also known as the style loss [4]. Similarly to Gatys et al. [3], Gram matrices were computed on layers *conv1_1*, *pool1*, *pool2*, *pool3*, and *pool4* of VGG-19.

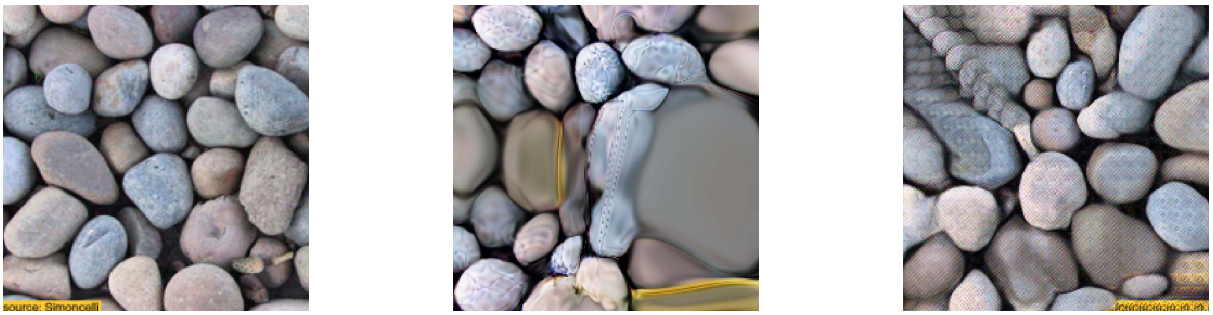

Figure 3: CPPNs vs. F-CPPNs for texture synthesis. (left) Target texture of pebbles. (middle) Output from Mordvintsev et al. [12]'s CPPN. (right) Output from our F-CPPN. By explicitly modelling frequencies beyond the DC component, our Fourier parameterization provides an improvement in surface detail on the synthesized pebbles.

In the case of Gatys et al. [3], textures were synthesized by directly optimizing their pixel values. Recent approaches use generative ConvNets to synthesize textures [7, 16], parameterizing the output by the ConvNet's weights. Our implementation follows a similar approach, however, we use a F-CPPN as the generative network. This is a novel application of CPPNs. Results are shown in Fig. 3. By explicitly modelling frequencies beyond the DC component, our F-CPPN provides an improvement in surface detail on the synthesized texture of pebbles. However, we observe a periodic tiling in the top-left region of the output. Whereas regions of constant output would correspond to untextured regions of constant colour in a CPPN (visible in CPPN output in Fig. 3), in a F-CPPN it would correspond to regions of the same coefficients, resulting in a $H_F \times W_F$-periodic tiling throughout the region.

4. Conclusion

In this paper, we presented an extension to CPPNs, based on Fourier analysis, which we call Fourier-CPPNs (F-CPPNs). F-CPPNs explicitly model the frequency information for each pixel output, capturing high-frequency detail that can not be captured by CPPNs. We applied our F-CPPN to the tasks of image reconstruction and texture synthesis and showed that the resulting images exhibited greater detail than the images synthesized by a CPPN. We observed a limitation common to both F-CPPNs and CPPNs where regions of constant output manifested themselves as regions of periodic tiling of texture in a F-CPPN's output and untextured regions of constant colour in a CPPN's output. Regularization methods may alleviate these issues, which we leave as directions for future work. An advantage of F-CPPNs is the direct manipulation of frequencies, allowing for interesting effects such as phase shifting [2] and band-pass filtering. We aim to explore these techniques in future work.

References

[1] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. 3

[2] W. T. Freeman, E. H. Adelson, and D. J. Heeger. Motion without movement. In *SIGGRAPH*, 1991. 4

[3] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NeurIPS*, 2015. 3, 4

[4] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 3

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 1

[6] D. Ha. Generating large images from latent vectors, 2016. 1

[7] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1, 4

[8] A. Karpathy. Image regression, 2014. 1

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018. 1

[10] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv*, 2018. 1

[11] L. Metz and I. Gulrajani. Compositional pattern producing GAN. In *NeurIPS Workshops*, 2017. 1

[12] A. Mordvintsev, N. Pezzotti, L. Schubert, and C. Olah. Differentiable image parameterizations. *Distill*, 2018. 1, 2, 3, 4

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 3

[14] X. Snelgrove and M. Tesfaldet. Interactive CPPNs in GLSL. In *NeurIPS Workshops*, 2018. 1

[15] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *GPEV*, 8(2):131–162, 2007. 1

[16] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 1, 4