Image Generators with Conditionally-Independent Pixel Synthesis

I. Anokhin[1,2], K. Demochkin[1,2], T. Khakhulin[1,2], G. Sterkin[1], V. Lempitsky[1,2], D. Korzhenkov[1]

[1]Samsung AI Center, Moscow
[2]Skolkovo Institute of Science and Technology, Moscow

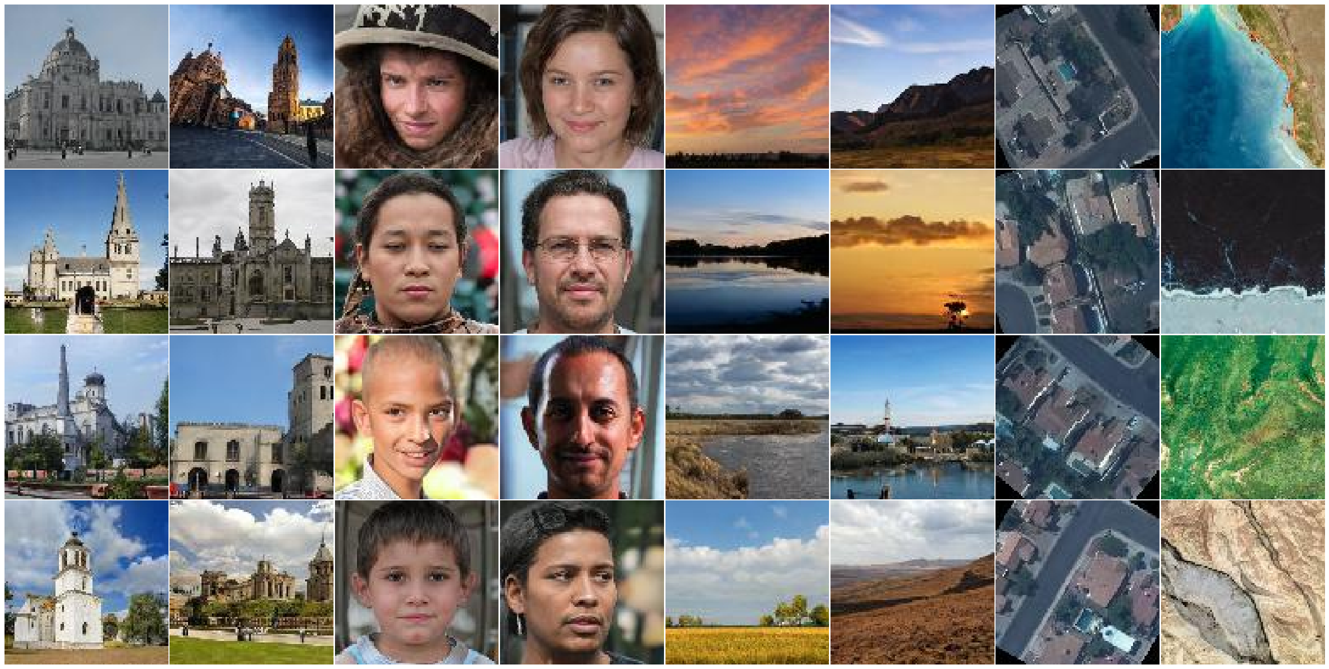


Figure 1: Samples from our generators trained on several challenging datasets (LSUN Churches, FFHQ, Landscapes, Satellite-Buildings, Satellite-Landscapes) at resolution 256×256. The images are generated without spatial convolutions, upsampling, or self-attention operations. No interaction between pixels takes place during inference.

Abstract

Existing image generator networks rely heavily on spatial convolutions and, optionally, self-attention blocks in order to gradually synthesize images in a coarse-to-fine manner. Here, we present a new architecture for image generators, where the color value at each pixel is computed independently given the value of a random latent vector and the coordinate of that pixel. No spatial convolutions or similar operations that propagate information across pixels are involved during the synthesis. We analyze the modeling capabilities of such generators when trained in an adversarial fashion, and observe the new generators to achieve similar generation quality to state-of-the-art convolutional generators. We also investigate several interesting properties unique to the new architecture.

1. Introduction

State-of-the-art in unconditional image generation is achieved using large-scale convolutional generators trained in an adversarial fashion [10, 11, 1]. While lots of nuances and ideas have contributed to the state-of-the-art recently, for many years since the introduction of DC-GAN [22] such generators are based around spatial convolutional layers, also occasionally using the spatial self-attention blocks [32]. Spatial convolutions are also invariably present in other popular generative architectures for images, including autoencoders [14], autoregressive generators [30], or flow models [3, 13]. Thus, it may seem that spatial convolutions (or at least spatial self-attention) is an unavoidable building block for state-of-the-art image generators.

Recently, a number of works have shown that individual images or collections of images of the same scene can be encoded/synthesized using rather different deep architectures (deep multi-layer perceptrons) of a special kind [20, 26]. Such architectures are not using spatial convolutions or spatial self-attention and yet are able to reproduce images rather well. They are, however, restricted to individual scenes. In this work, we investigate whether deep generators for unconditional image class synthesis can be built using similar architectural ideas, and, more importantly, whether the quality of such generators can be pushed to state-of-the-art.

Perhaps surprisingly, we come up with a positive answer (Fig. 1), at least for the medium image resolution (of 256×256). We have thus designed and trained deep generative architectures for diverse classes of images that achieve similar quality of generation to state-of-the-art convolutional generator StyleGANv2 [11], even surpassing this quality for some datasets. Crucially, our generators are not using any form of spatial convolutions or spatial attention in their pathway. Instead, they use coordinate encodings of individual pixels, as well as sidewise multiplicative conditioning (weight modulation) on random vectors. Aside from such conditioning, the color of each pixel in our architecture is predicted independently (hence we call our image generator architecture *Conditionally-Independent Pixel Synthesis* (CIPS) generators).

In addition to suggesting this class of image generators and comparing its quality with state-of-the-art convolutional generators, we also investigate the extra flexibility that is permitted by the independent processing of pixels. This includes easy extention of synthesis to non-trivial topologies (e.g. cylindrical panoramas), for which the extension of spatial convolutions is known to be non-trivial [17, 2]. Furthermore, the fact that pixels are synthesized independently within our generators, allows sequential synthesis for memory-constrained computing architectures. It enables our model to both improve the quality of photos and generate more pixel values in a specific areas of image (i.e. to perform foveated synthesis).

2. Related Work

Feeding pixel coordinates as an additional input to the neural network previously was successfully used in the widely known CoordConv technique [18] to introduce the spatial-relational bias. Recently, the same idea was employed by the COCO-GAN [17] to generate images by parts or create "looped" images like spherical panoramas. However, those models still used standard convolutions as the main synthesis operation. The synthesis process for neighboring pixels in such architectures is therefore not independent.

To the best of our knowledge, the problem of regressing a given image from pixel coordinates with a perceptron (that calculates each pixel's value independently) started from creating compositional patterns with an evolutionary approach [28]. Those patterns, appealing for digital artists, were also treated as kind of differentiable image parametrization [21]. However, this approach was not capable of producing photorealistic hi-res outputs (e.g., see the demo [8]).

Some machine learning blogs reported experiments with GANs, where the generator was a perceptron that took a random vector and pixel coordinates as an input, and returned that pixel's value as an output [5, 6]. The described model was successfully trained on MNIST, but has not been scaled to more complex image data.

Scene-representation networks [27] and later the neural radiance fields (NeRF) networks [20] have demonstrated how 3D content of individual scenes can be encoded with surprising accuracy using deep perceptron networks. Following this realization, systems [26] and [29] considered the usage of periodic activation functions and so-called Fourier features to encode the pixel (or voxel) coordinates, fed to the multi-layer perceptron. In particular, the ability to encode high-resolution individual images in this way was demonstrated. All these works however have not considered the task of learning image generators, which we address here.

The very recent (and independent) Generative Radiance Fields (GRAF) system [25] showed promising results at embedding the NeRF generator into an image generator for 3D aware image synthesis. Results for such 3D aware synthesis (still limited in diversity and resolution) for certain have been demonstrated. Here, we do not consider 3D-aware synthesis and instead investigate whether perceptron-based architectures can achieve high 2D image synthesis quality.

3. Method

Our generator network synthesizes images of a fixed resolution $H \times W$ and has the multi-layer perceptron-type architecture G (see Fig. 2). In more detail, the synthesis of each pixel takes a random vector $\mathbf{z} \in \mathcal{Z}$ shared across all pixels, as well the pixel coordinates $(x, y) \in \{0 \dots W-1\} \times \{0 \dots H-1\}$ as input. It then returns the RGB value $\mathbf{c} \in [0,1]^3$ of that pixel $G : (x, y, \mathbf{z}) \mapsto \mathbf{c}$. Therefore, to compute the whole output image I, the generator G is evaluated at at each pair (x, y) of the coordinate grid, while keeping the random part $\mathbf{z}$ fixed:

$$I = \{G(x, y; \mathbf{z}) \mid (x, y) \in \mathtt{mgrid}(H, W)\}, \qquad (1)$$

where

$$\mathtt{mgrid}(H, W) = \{(x, y) \mid 0 \le x < W, 0 \le y < H\}$$

is a set of integer pixel coordinates.

Following [10], a mapping network M (also a perceptron) turns $\mathbf{z}$ into a *style* vector $\mathbf{w} \in \mathcal{W}$, $M : \mathbf{z} \mapsto \mathbf{w}$, and all the stochasticity in the generating process comes from this style component.

We then follow the StyleGANv2 [11] approach of injecting the style $\mathbf{w}$ into the process of generation via weight modulation. To make the paper self-contained, we describe the procedure in brief here.

Any modulated fully-connected (ModFC) layer of our generator (see Fig. 2) can be written in the form $\psi = \hat{B}\phi + \mathbf{b}$, where $\phi \in \mathbb{R}^n$ is an input, $\hat{B}$ is a learnable weight matrix $B \in \mathbb{R}^{m\times n}$ modulated with the style $\mathbf{w}$, $\mathbf{b} \in \mathbb{R}^m$ is a learnable bias, and $\psi \in \mathbb{R}^m$ is an output. The modulation takes place as follows: at first, the style vector $\mathbf{w}$ is mapped with a small net (referred to as A in Fig. 2) to a scale vector $\mathbf{s} \in \mathbb{R}^n$ Then, the (i, j)-th entry of $\hat{B}$ is computed as

$$\hat{B}_{ij} = \frac{s_j B_{ij}}{\sqrt{\epsilon + \sum\limits_{k=1}^{n} (s_k B_{ik})^2}}, \quad (2)$$

where ϵ is a small constant. After this linear mapping, a LeakyReLU function is applied to ψ.

Finally, in our default configuration we add skip connections for every two layers from intermediate feature maps to RGB values and sum the contributions of RGB outputs corresponding to different layers. These skip connections naturally add values corresponding to the same pixel, and do not introduce interactions between pixels.

We note that the independence of the pixel generation process, makes our model parallelizable at inference time and, additionally, provides flexibility in the latent space z. E.g., as we show below, in some modified variants of synthesis, each pixel can be computed with a different noise vector z, though gradual variation in z is needed to achieve consistently looking images.

3.1. Positional encoding

The architecture described above needs an important modification in order to achieve the state-of-the-art synthesis quality. Recently two slightly different versions of positional encoding for coordinate-based multi-layer perceptrons (MLP), producing images, were described in literature. Firstly, SIREN [26] proposed a perceptron with a principled weight initialization and sine as an activation function, used throughout all the layers. Secondly, the Fourier features, introduced in [29], employed a periodic activation function in the very first layer only. In our experiments, we apply a somewhat in-between scheme: the sine function is used to obtain Fourier embedding e_{fo}, while other layers use a standard LeakyReLU function:

$$e_{fo}(x, y) = \sin\left[B_{fo}(x', y')^T\right], \quad (3)$$

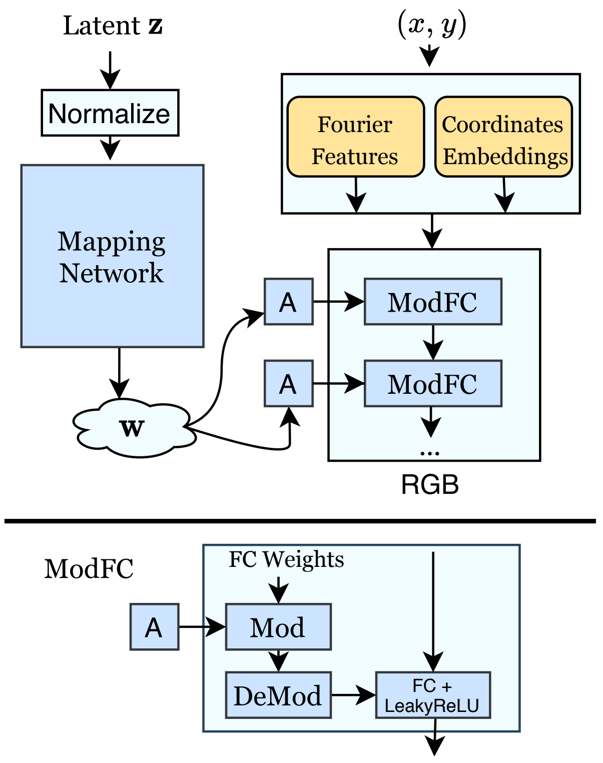


Figure 2: The Conditionally-Independent Pixel Synthesis (CIPS) generator architecture. Top: the generation pipeline, in which the coordinates (x, y) of each pixel are encoded (yellow) and processed by a fully-connected (FC) network with weights, modulated with a latent vector $\mathbf{w}$, shared for all pixels. The network returns the RGB value of that pixel. Bottom: The architecture of a modulated fully-connected layer (ModFC). Note: our default configuration also includes skip connections to the output (not shown here).

where $x' = \frac{2x}{W-1} - 1$ and $y' = \frac{2y}{H-1} - 1$ are pixel coordinates, uniformly mapped to the range $[-1, 1]$ and the weight matrix $B_{fo} \in \mathbb{R}^{2\times n}$ is learnable, like in SIREN paper.

However, only Fourier positional encoding usage turned out insufficient to produce plausible images. In particular, we have found out that the outputs of the synthesis tend to have multiple wave-like artifacts. Therefore, we also train a separate vector $e_{co}^{(x,y)}$ for each spatial position and call them *coordinate embeddings*. They represent $H \times W$ learnable vectors in total. For comparison of these two embedding from the spectral point of view, see Sec. 4.5. The full positional encoding $e(x, y)$ is a concatenation of Fourier features and a coordinate embedding

$$e(x, y) = \text{concat}\left[e_{fo}(x, y), e_{co}^{(x,y)}\right] \quad (4)$$

and serve as an input for the next perceptron layer: $G(x, y; \mathbf{z}) = G'(e(x, y); M(\mathbf{z}))$.

4. Experiments

4.1. Architecture details

In our experiments, both Fourier features and coordinate embeddings had the dimension of 512. The generator had 14 modulated fully-connected layers of width 512 We use leaky ReLU activation with the slope 0.2. We implement our experiments on top of the public code[1] for StyleGANv2. Our model is trained with a standard non-saturating logistic GAN loss with R_1 penalty [19] applied to the discriminator D. The discriminator has a residual architecture, described in [11] (we have deliberately kept the discriminator architecture intact). Networks were trained by Adam optimizer [12] with learning rate 2×10^{-3} and hyperparameters: $\beta_0 = 0$, $\beta_1 = 0.99$, $\epsilon = 10^{-8}$.

4.2. Evaluation

We now evaluate CIPS generators and their variations on a range of datasets. For the sake of efficiency, most evaluations are restricted to 256×256 resolution. The following datasets were considered:

- The *Flickr Faces-HQ* (FFHQ) [10] dataset contains 70,000 high quality well-aligned, mostly near frontal human faces. This dataset is the most regular in terms of geometric alignement and the StyleGAN variants are known to perform very well in this setting.

- The *LSUN Churches* [31] contains 126,000 outdoor photographs of churches of rather diverse architectural style. The dataset is regular, yet images all share upright orientation.

- The *Landscapes* dataset contains 60,000 manually collected landscape photos from the Flickr website.

- The *Satellite-Buildings*[2] dataset contains 280,741 images of 300×300 pixels (which we crop to 256×256 resolution and randomly rotate). This dataset has large size, and is approximately aligned in terms of scale, yet lacks consistent orientation.

- Finally, the *Satellite-Landscapes*[3] contains a smaller curated collection of 2,608 images of 512×512 resolution of satellite images depicting various impressive landscapes found on Google Earth (which we crop to 256×256 resolution). This is the most "textural" dataset, that lacks consistent scale or orientation.

For evaluation, we relied on commonly used metrics for image generation: Frechet Inception Distance (FID) [7] as well as more recently introduced generative Precision and Recall measures [23, 15].

[1]https://github.com/rosinality/stylegan2-pytorch
[2]https://www.crowdai.org/challenges/mapping-challenge
[3]https://earthview.withgoogle.com/

	StyleGANv2	CIPS (ours)
FFHQ	3.83	4.38
LSUN Churches	3.86	2.92
Landscapes	2.36	3.61
Satellite-Buildings	73.67	69.67
Satellite-Landscapes	51.54	48.47

Table 1: FID on multiple datasets at resolution of 256^2 for CIPS-skips model. Note that CIPS is of comparable quality with state-of-the-art StyleGANv2, and better on Churches. The value for CIPS model on FFHQ differs from the one reported in Tab. 3 as we trained this model for more time and with larger batch size.

Model	Precision	Recall
StyleGANv2	0.609	0.513
CIPS	0.613	0.493

Table 2: Precision & Recall measured on FFHQ at 256^2. The resulting quality of our model is better in terms of precision (corresponds to plausibility of images) and worse in recall (this points to the greater number of dropped modes).

Our main evaluation is thus against the state-of-the-art StyleGANv2 generator [11]. We took FID value for LSUN Churches directly from the original paper [11] and trained StyleGANv2 on other datasets in authors' setup but without style-mixing and path regularization of generator – as noted in the original paper, these changes do not influence the FID metric. The results of this key comparison are presented in Tab. 1 and 2. Neither of the two variants of the generator dominates the other, with StyleGANv2 achieving lower (better) FID score on FFHQ and Landscapes, while CIPS generator achieving lower score on LSUN Churches and both Satellite datasets.

4.3. Ablations

We then evaluate the importance of different parts of our model by its ablation on the FFHQ dataset (Tab. 3). We thus consider removing Fourier features, coordinate embeddings (config referred to as *CIPS-NE*) and replace LeakyReLU activation with sine function in all layers. We also compare the variants with residual connections (we follow StyleGANv2 [11] implementation adjusting variance of residual blocks with the division by $\sqrt{2}$) with our main choice of cumulative projections to RGB. Additionally, the "base" configuration without skip connections and residual connections is considered. In this comparison, all models were trained for 300K iterations with batch size of 16.

As the results show, coordinate embeddings, residual blocks and cumulative projection to RGB significantly im-

CIPS	"base"	"No embed (NE)"	"No Fourier"	**Main**	"Residual"	Sine
Fourier Features	+	+	–	+	+	–
Coordinate Embeddings	+	–	+	+	+	+
Residual blocks	–	–	–	–	+	–
Skip connections	–	–	–	+	–	–
Sine Activation	–	–	–	–	–	+
FID	6.71	12.71	10.18	**6.31**	6.52	10.0

Table 3: Effects of the modifications of CIPS generator on the FFHQ dataset in terms of Frechet Inception Distance (FID) score. Each column corresponds to a certain configuration, while rows correspond to present/missing features. The simultaneous usage of Fourier features and coordinate embeddings is necessary for a good FID score. Also, both residual connections and cumulative skip connections (default configuration) to the output outperform the plain multilayer perceptron.

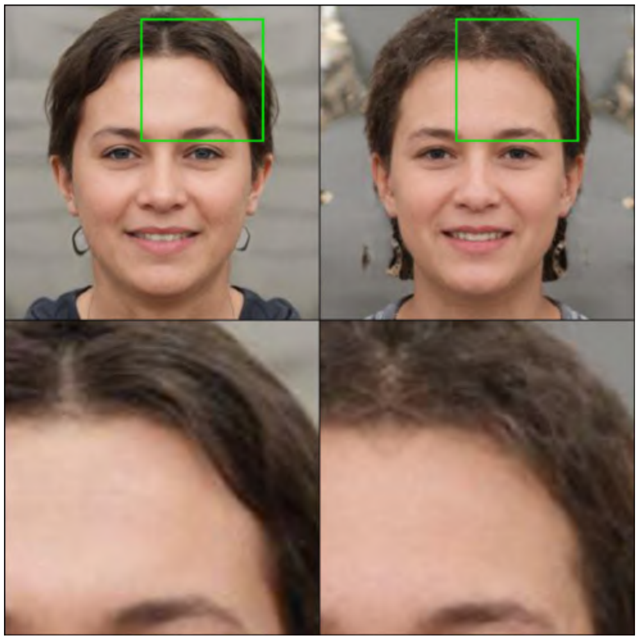
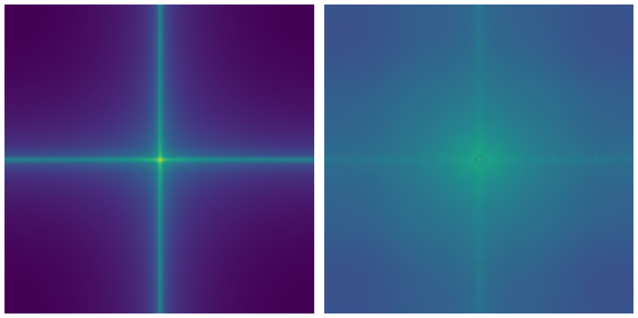


Figure 3: Image corresponding to the mean style vector in the space of $\mathcal{W}$ for CIPS (left) and CIPS-NE (no embeddings) generators (right). Left image has more plausible details like hair which confirms the results in Tab. 3.

prove the quality of the model. The removal of coordinate embeddings most severely worsens the FID value, and affects the quality of generated images (Fig. 3). We further investigate the importance of coordinate embeddings for the CIPS model below.

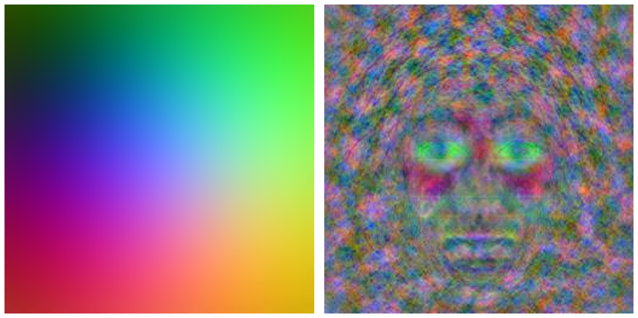
4.4. Influence of positional encodings

To analyze the difference between Fourier features e_{fo} and coordinate embeddings e_{co}, we plotted the spectrum of these codes for the generator CIPS-base, trained on FFHQ. As shown in Fig. 4, Fourier encoding generally carries low-frequency components, whereas coordinate embeddings resemble more high-frequency details. The Principal Component Analysis (PCA) of the two encodings supports the same conclusion (Fig. 5) The possible explanation is simple: coordinate embeddings are trained independently for each pixel, while $e_{fo}(x, y)$ is a learned function of the coordinates. However, the next layers of the network could



(a) Fourier features (b) Coordinate embeddings

Figure 4: The spectrum magnitude for our two kinds of positional encoding (color scale is equal for both plots). The output of coordinate embeddings clearly has more higher frequencies.



(a) Fourier features (b) Coordinate embeddings

Figure 5: PCA plot (3 components) for two kinds of positional encoding of CIPS-base. Coordinate embeddings contain not just more fine-grained details, but also key points of the averaged face.

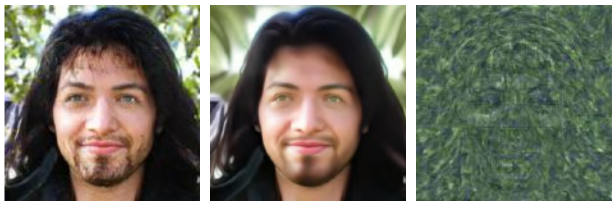


Figure 6: Influence of different types of positional encoding on the resulting image. Left: original image. Center: coordinate embeddings zeroed out (the image contains no fine-grained details). Right: Fourier features zeroed out (only high-frequency details are present).

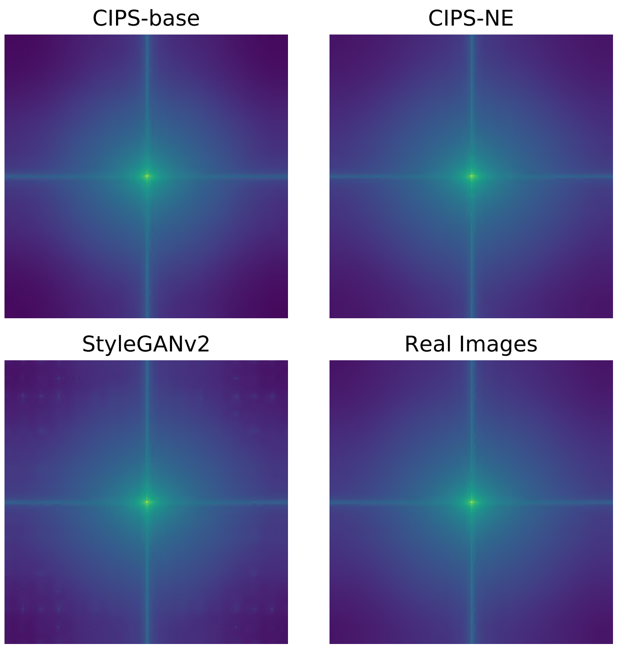
transform the positional codes and, for example, finally produce more fine-grained details, relying on Fourier features. To demonstrate that this is not the case, we conducted the following experiment. We have zeroed out either the output of Fourier features or coordinate embeddings and showed the obtained images in Fig. 6. One can notice that the information about the facial hair's details as well as the forelock is located in the coordinate embeddings. This proves that it is the coordinate embeddings that are the key to high-frequency details of the resulting image.

4.5. Spectral analysis of generated images

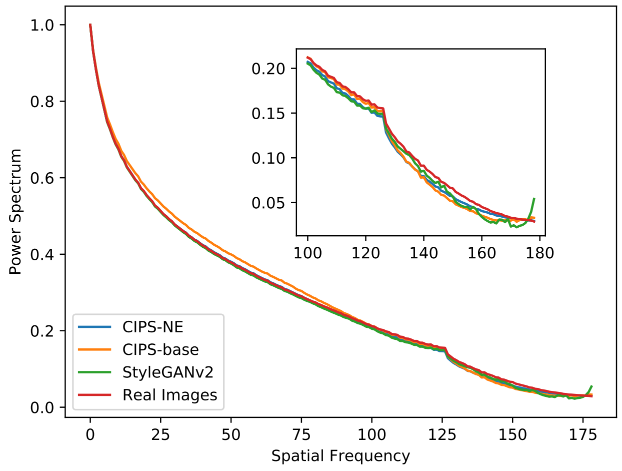
Recently, [4] observed that the common convolutional upsampling operations can lead to the inability to learn the spectral distribution of real images, in spite of any generator architecture. In contrast, CIPS operates explicitly with the coordinate grid and has no upscaling modules, which should to improved reproduction of the spectrum. Indeed, we compare the spectrum of our models (CIPS-"base" without residual and skip connections; CIPS-NE) to StyleGANv2 and demonstrate that CIPS generators design has the advantage in the spectral domain.

The analysis of magnitude spectra for produced images is given in Fig. 7a. The spectrum of StyleGANv2 has artifacts in high-frequency regions, not present in both CIPS generators under consideration. Following prior works [4], we also use the azimuthal integration (AI) over the Fourier power spectrum (Fig. 7b). It is worth noting that AI statistics of CIPS-NE are very close to the ones of real images. However, adding the coordinate embeddings degrades a realistic spectrum while improving the quality in terms of FID (Tab. 3).

We note that the introduction of skip connections in fact makes the spectra less similar to those of natural images.



(a) Magnitude spectrum. Our models produce less artifacts in high frequency components (note the grid-like pattern in StyleGANv2). Two CIPS models are difficult to distinguish between (better zoom in).



(b) Azimuthal integration over Fourier power spectrum. The curve of StyleGANv2 has heavy distortions in most high frequency components. Surprisingly, CIPS-NE demonstrates a more realistic and smooth tail than CIPS-base, while being worse in terms of FID.

Figure 7: Spectral analysis for models trained on FFHQ at resolution of 256^2. All results are averaged across 5000 samples. We demonstrate that CIPS-NE is most similar to real images.

4.6. Interpolation

We conclude the experimental part with the demonstration of the flexibility of CIPS. As well as many other generators, CIPS generators have the ability to interpolate between latent vectors with meaningful morphing (Fig. 10). As expected, the change between the extreme images occurs smoothly and allows for the use of this property, in a

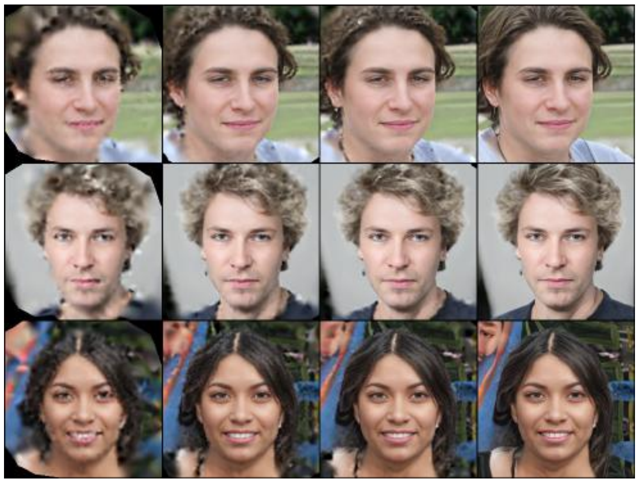


Figure 8: Images generated using foveated synthesis. In each case, the CIPS generator was sampled on a 2D Gaussian distribution concentrated in the center of an image (standard deviation = $0.4*$image size). Left to right: sampled pattern covers 5% of all pixels, 25%, 50%, 100% (full coordinate grid). Missing color values have been filled via bicubic interpolation.

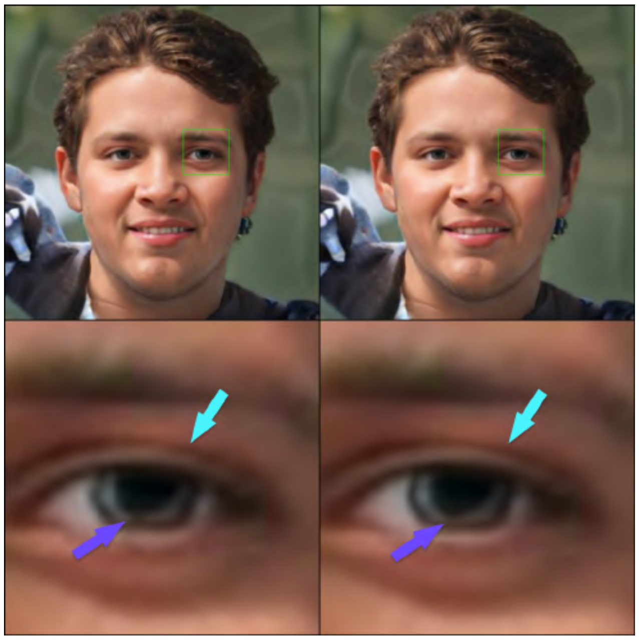


Figure 9: Left: the generated image of resolution 256×256, upscaled with Lanczos upsampling scheme [16] to 1024×1024. Right: the image, synthesized by CIPS, trained at resolution of 256×256 on the coordinate grid of resolution 1024×1024. Note the sharpness/plausibility of the eyelid and the more proper shape of the pupil.

similar vein as in the original works (e.g. [11]).

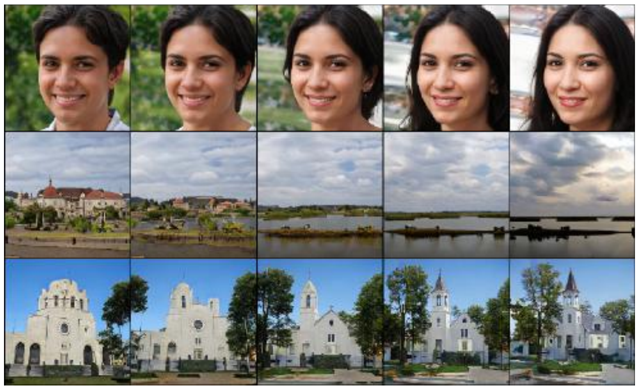


Figure 10: Latent linear morphing between two sampled images – the left-most and right-most ones.

4.7. Foveated rendering and interpolation

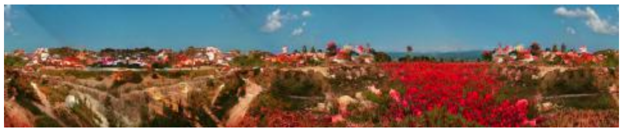
One of the inspiring applications of our per-pixel generator is the foveated synthesis. The foveated synthesis ability can be beneficial for computer graphics and other applications, as well as mimics human visual system. In foveated synthesis, an irregular grid of coordinates is sampled first, more dense in the area, where the gaze is assumed to be directed to, and more sparse outside of that region. After that, CIPS is evaluated on this grid (its size is less than the full resolution), and color for missing pixels of the image is filled using interpolation. The demonstration of this method is provided in Fig. 8.

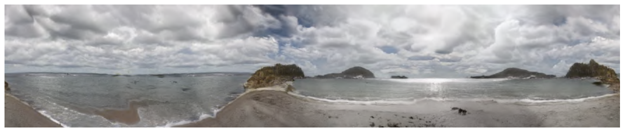
Alongside the foveated rendering, we are also able to interpolate the image beyond the training resolution by simply sampling denser grids. Here we use a model, trained on images of 256×256 resolution to process a grid of 1024×1024 pixels and compare it with upsampling the results of upsampling the image synthesized at the 256×256 resolution with the Lanczos filter [16]. As Fig. 9 suggests, more plausible details are obtained with denser synthesis than with Lanczos filter.

4.8. Panorama synthesis

As CIPS is built upon a coordinate grid, it can relatively easily use non-Cartesian grids. To show this, we thus adopt a cylindrical system to produce landscape panoramas. The training setup is as follows. We uniformly sample a crop 256×256 from the cylindrical coordinate grid and train the generator to produce images using these coordinate crops as inputs. A similar idea was also explored in [17]. We note, however, that during training we do not use any real panoramas in contrast to other coordinate-based COCO-GAN model [17]. Fig. 11a and 11b provide examples of panorama samples obtained with the resulting model.

As each pixel is generated from its coordinates and style vector only, our architecture admits pixel-wise style inter-


(a)


(b)

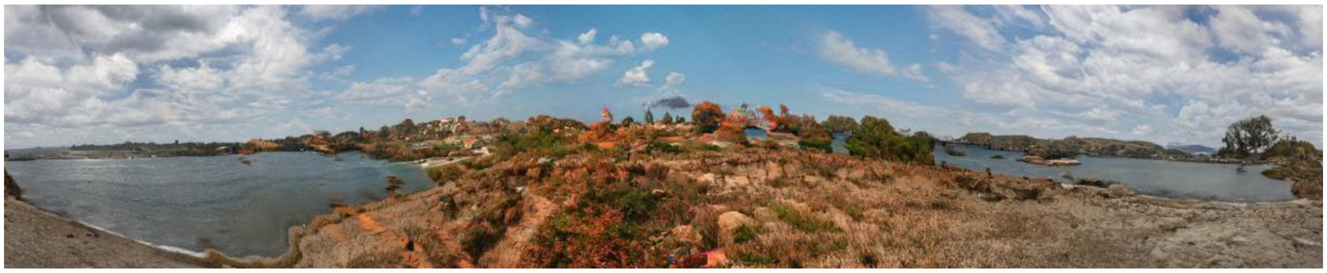

(c)

Figure 11: Panorama blending. We linearly blend two upper images from CISP generator trained on the Landscapes dataset with a cylindrical coordinate system. The resulting image contains elements from both original panoramas: land and water integrated naturally.

polation (Fig. 11c). In these examples, the style vector blends between the central part (the style of Fig. 11a) and the outer part (the style of 11b).

4.9. Typical artifacts

Finally, we show the typical artifacts that keep recurring in the results of CIPS generators (Fig. 12). We attribute the wavy texture (in hair) and repeated lines pattern (in buildings) to the periodic nature of sine activation function within the Fourier features. Also we note that sometimes CIPS produces a realistic image with a small part of the image being inconsistent with the rest and out of the domain. Our belief is that this behaviour is caused by the LeakyReLU activation function that divides the coordinate grid into parts. For each part, CIPS effectively applies its own inverse discrete Fourier transform. As CIPS generators do not use any upsampling or other pixel coordination, it is harder for the generator to safeguard against such behaviour.

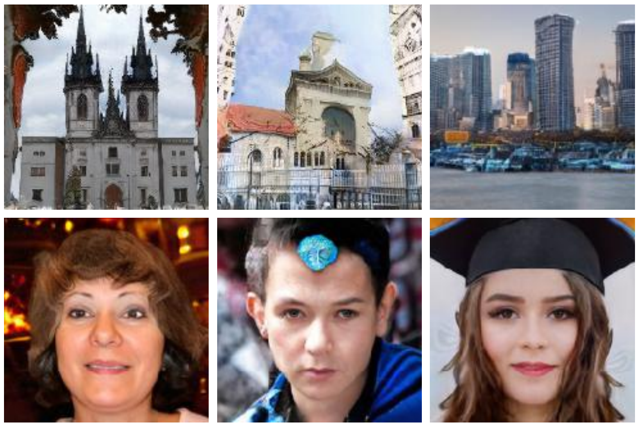


Figure 12: Examples of the most common kinds of artifacts on different datasets. They are best described as wavy textures on hair, background, and glowing blobs. see text for the discussion.

5. Conclusion

We presented a new generator model called CIPS, a high-quality architecture with conditionally independent pixel synthesis, such that the color value is computed using only random noise and coordinate position.

Our key insight is that the proposed architecture without spatial convolutions, attention or upsampling operations has the ability to compete in the model market and obtain decent quality in terms of FID and precision & recall; such results have not been presented earlier for perceptron-based models. Furthermore, in the spectral domain outputs of CIPS are harder to discriminate from real images. Interestingly, CIPS-NE modification is weaker in terms of plausibility, yet has a more realistic spectrum.

Direct usage of a coordinate grid allows us to work with more complex structures, such as cylindrical panoramas, just by replacing the underlying coordinate system.

In summary, our generator demonstrates quality on par with state-of-the-art model StyleGANv2; moreover, it has applications in various diverse scenarios. We have shown that the considered model could be successfully applied to foveated rendering and super-resolution problems in their generative interpretations. Future development of our approach assumes researching these problems in their image-to-image formulations.

References

[1] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. 1

[2] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018. 2

[3] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 1

[4] R. Durall, M. Keuper, and J. Keuper. Watch Your Up-Convolution: CNN Based Generative Deep Neural Networks Are Failing to Reproduce Spectral Distributions. In *Proc. CVPR*, pages 7887–7896, 2020. 6

[5] D. Ha. Generating large images from latent vectors. *blog.otoro.net*, 2016. 2

[6] D. Ha. Generating large images from latent vectors - part two. *blog.otoro.net*, 2016. 2

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proc. NIPS*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. 4

[8] A. Karpathy. Convnetjs demo: Image "painting". https://cs.stanford.edu/people/karpathy/convnetjs/demo/image_regression.html. Accessed: 2020-11-05. 2

[9] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training Generative Adversarial Networks with Limited Data. In *Proc. NeurIPS*. Curran Associates, Inc., 2020. 12

[10] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proc. CVPR*, pages 4396–4405, 2019. 1, 3, 4, 12

[11] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proc. CVPR*, pages 8107–8116, 2020. 1, 2, 3, 4, 7

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015. 4

[13] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Proc. NeurIPS*, pages 10215–10224, 2018. 1

[14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[15] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Proc. NeurIPS*, volume 32, pages 3927–3936. Curran Associates, Inc., 2019. 4

[16] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950. 7

[17] C. H. Lin, C. Chang, Y. Chen, D. Juan, W. Wei, and H. Chen. Coco-gan: Generation by parts via conditional coordinating. In *Proc. ICCV*, pages 4511–4520, 2019. 2, 7

[18] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Proc. NeurIPS*, pages 9627–9638. Curran Associates, Inc., 2018. 2

[19] L. Mescheder, A. Geiger, and S. Nowozin. Which Training Methods for GANs do actually Converge? In J. Dy and A. Krause, editors, *Proc. ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3481–3490, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR. 4

[20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Proc. ECCV*, pages 405–421, Cham, 2020. Springer International Publishing. 2

[21] A. Mordvintsev, N. Pezzotti, L. Schubert, and C. Olah. Differentiable image parameterizations. *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations. 2

[22] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016. 1

[23] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Proc. NIPS*, volume 31, pages 5228–5237. Curran Associates, Inc., 2018. 4

[24] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. 12, 14

[25] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis. In *Proc. NeurIPS*. Curran Associates, Inc., 2020. 2, 11

[26] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS*. Curran Associates, Inc., 2020. 2, 3

[27] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Proc. NeurIPS*. 2019. 2

[28] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, Jun 2007. 2

[29] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Proc. NeurIPS*. Curran Associates, Inc., 2020. 2, 3

[30] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proc. ICML*, pages 1747–1756, 2016. 1

[31] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. 2016. 4

[32] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *Proc. ICML*, 2019. 1

[33] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016. 12

[34] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable Augmentation for Data-Efficient GAN Training. In *Proc. NeurIPS*. Curran Associates, Inc., 2020. 12

Appendices

A. Architecture details

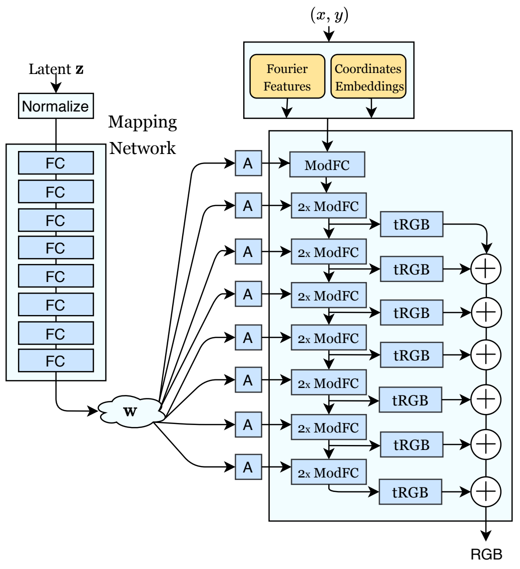


Figure 13: The diagram of the CIPS generator (default version).

Modification	# parameters (mln)
CIPS-“base”	43.8
CIPS-NE	10.2
CIPS-Default	45.9
StyleGANv2	30.0

Table 4: The number of parameters for different version of the CIPS generator. For reference, the number of parameters within the StyleGANv2 generator is also given.

In this section we provide additional information about the default version of our CIPS generator (Fig. 13). In total, its backbone contains 15 fully connected layers. The first layer projects concatenated coordinate embeddings and Fourier features into the joint space with the dimension of 512. Next, the following layer pattern is repeated seven times. The representation is put through two modulated fully-connected layers and a projection to RGB color space is computed. The projections coming from the seven iterations are summed together to create the final image. The number of parameters for the different modifications of the CIPS generator discussed in the paper are given in Tab. 4.

B. Coordinate embeddings

We also run the Principle Components Analysis (PCA) for coordinate embeddings of models trained on Landscapes and LSUN-Churches images (similar pattern for the FFHQ dataset is shown in the main paper). Fig. 14 provides the visualisation for the three main components. Note, that as these datasets are as aligned as FFHQ, there is considerably less spatial structural information in the learned embeddings.

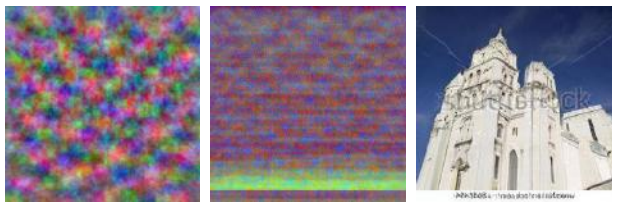


Figure 14: Visualisation of three main principal components of coordinate embeddings for CIPS models, trained on Landscapes (left) and LSUN-Churches (center). As these datasets are not as aligned as the face dataset, there is less recognizable structure in the learned coordinate embeddings. The bottom horizontal structure in the LSUN-Churches case is likely due to frequent watermark pattern in the dataset (a sample from the model with such watermark is shown on the right).

Dataset	Patch size	FID
FFHQ	256	4.38
	128	9.08
	64	11.79
LSUN-Churches	256	2.92
	128	7.08
	64	11.53

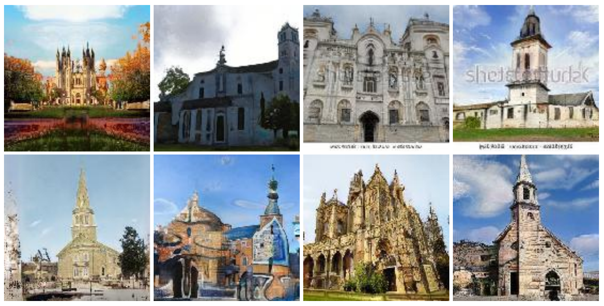
Table 5: Frechet Inception Distance (FID) values for CIPS models trained on patches of varying receptive field and fixed resolution (64×64 and 128×128). The results for patch-based training are worse than the default training procedure, in which the discriminator observes the full 256×256 image.

C. Patch-based generation

To show one benefit of coordinate-based approach, we demonstrate the results of *memory-constrained* training, where the discriminator observes patches at lower resolution than the full image (inspired by the GRAF system [25]). Since pixel generation is conditionally-independent, at each iteration only low-resolution patches need to be generated. Thus, only the following $K \times K$ patch is synthesized and submitted to the discriminator:

$$P_{K,\sigma}(u, v) = \{G(u + i\sigma, v + j\sigma; \mathbf{z}) \mid (i, j) \in \mathtt{mgrid}(K, K)\},$$

where $0 \le u < W - (K-1)\sigma$ and $0 \le v < H - (K-1)\sigma$ are the coordinates of the corner pixel of the patch. For $\sigma = 1$ this produces dense patch, while for


LSUN-Churches

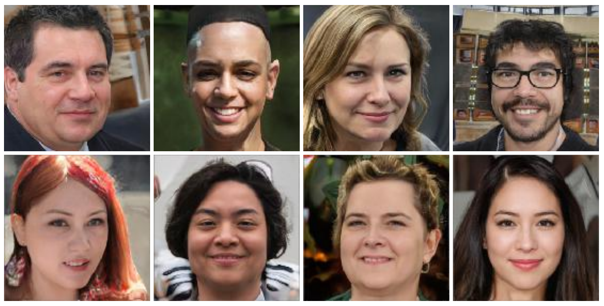

FFHQ

Figure 15: Samples from CIPS generators learned with memory-constrained **patch-based training**. Within every grid, the top row contains images from models trained with patches of size 128 $\times$ 128 and the bottom row represents outputs from training on 64 $\times$ 64 patches. While the samples obtained with such memory-constrained training are meaningful, their quality and diversity are worse compared to standard training.

$\sigma > 1$ dilated patch with increased receptive field is obtained. Applying this patch sampling to real images before putting them into the discriminator may be thought of as an example of a differentiable augmentation, the usefulness of which was recently proved by [9, 34].

Tab. 5 reports the quality (FID) for CIPS generators trained on patches of sizes 64×64 and 128×128, while the resolution of full images equals 256×256. Fig. 15 shows the outputs of models, trained with the patch-based pipeline. In our experiments, training with smaller size of patches degrades the overall quality of resulting samples.

D. Additional samples

In Fig. 16, we provide additional samples from CIPS generators trained on different datasets. We also demonstrate more samples of cylindrical panoramas in Fig. 17.

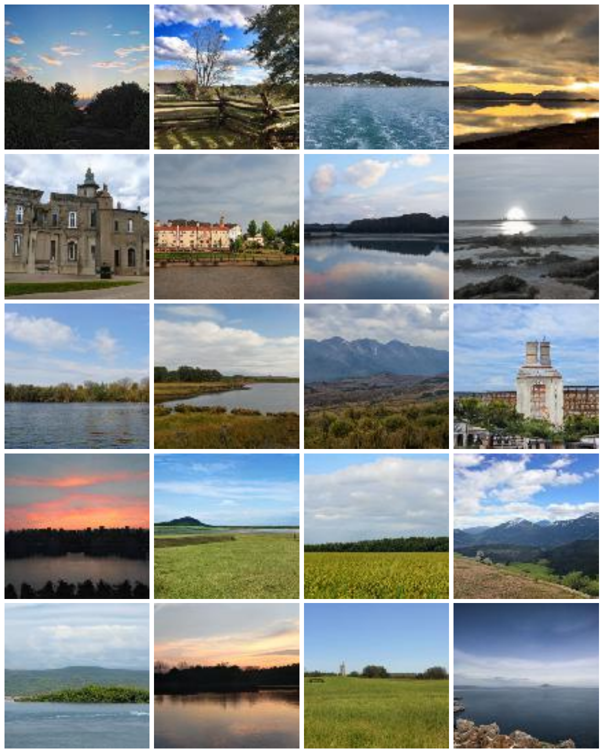
Although we do not apply mixing regularization [10] at train time, our model is still capable of layer-wise combination of latent variables at various depth (see Fig. 19). The examples suggest that similarly to StyleGAN, different layers of CIPS control different aspects of images.

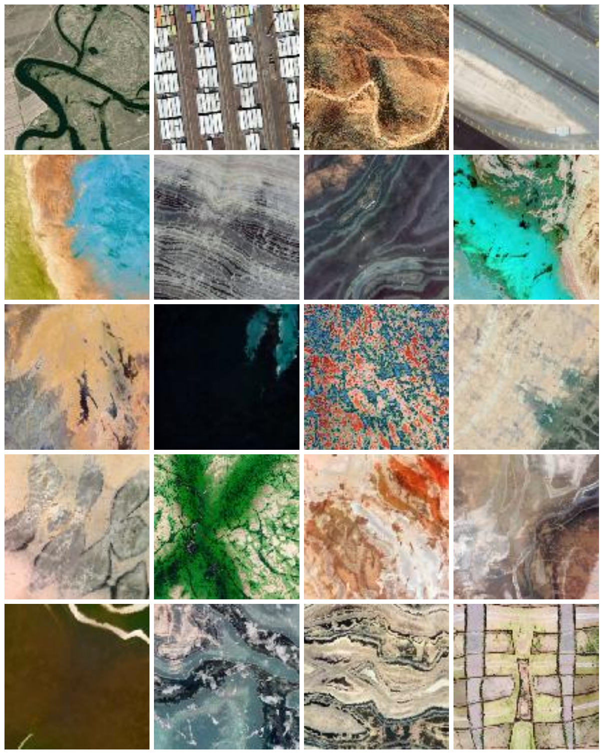
E. Nearest neighbors

To assess the generalization ability of CIPS architecture, we also show the samples from the model trained on the FFHQ face dataset alongside the most similar faces from the train dataset. To mine the most similar faces, we extract faces using the MTCNN model [33], and then compute their embeddings using FaceNet [24] (the public implementation of these models[4] was used). Fig. 18 shows five nearest neighbors (w.r.t. FaceNet descriptors) for each samples. The samples generated by the model are clearly not duplicates of the training images.

[4]https://github.com/timesler/facenet-pytorch

LSUN-Churches


Landscapes


Satellite-Landscapes

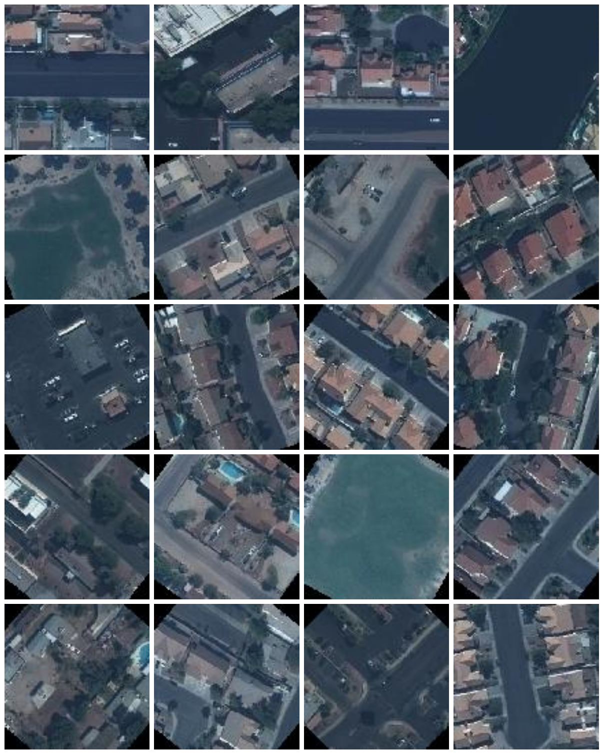

Satellite-Buildings

Figure 16: Samples from CIPS generators trained on various datasets. The top row of every grid shows real samples, and the remaining rows contain samples from the models. The samples from CIPS generators are plausible and diverse.

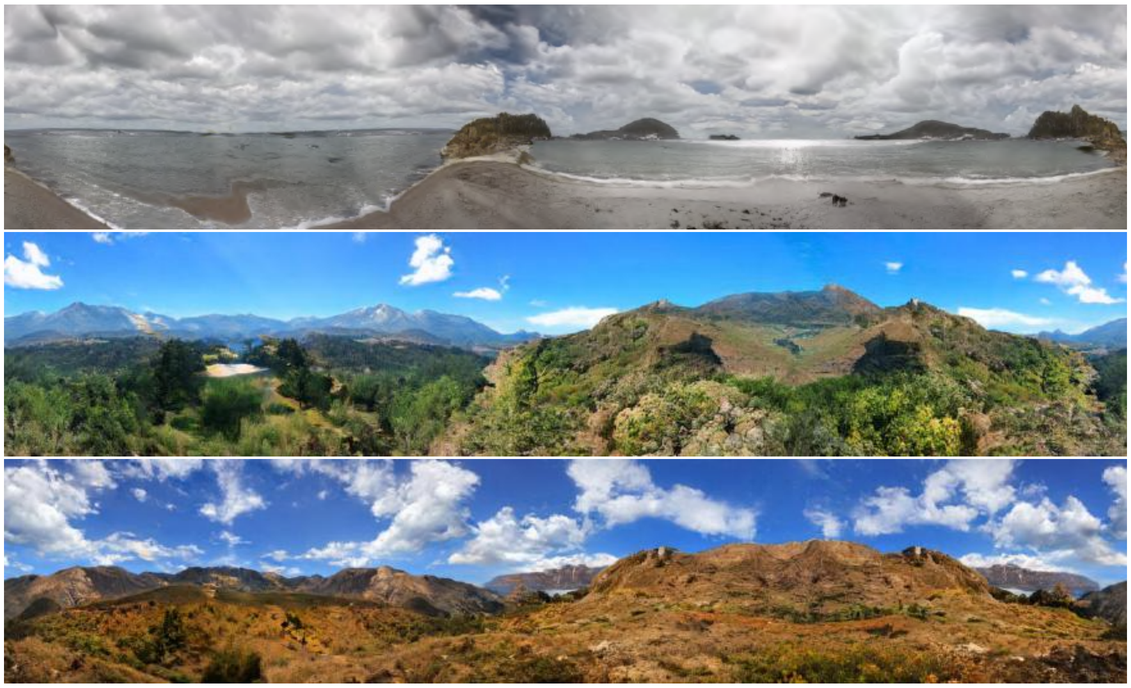


Figure 17: Additional samples of cylindrical panoramas, generated by the CIPS model trained on the Landscapes dataset. The training data contains standard landscape photographs from the Flickr website. No panoramas are provided to the model during training.

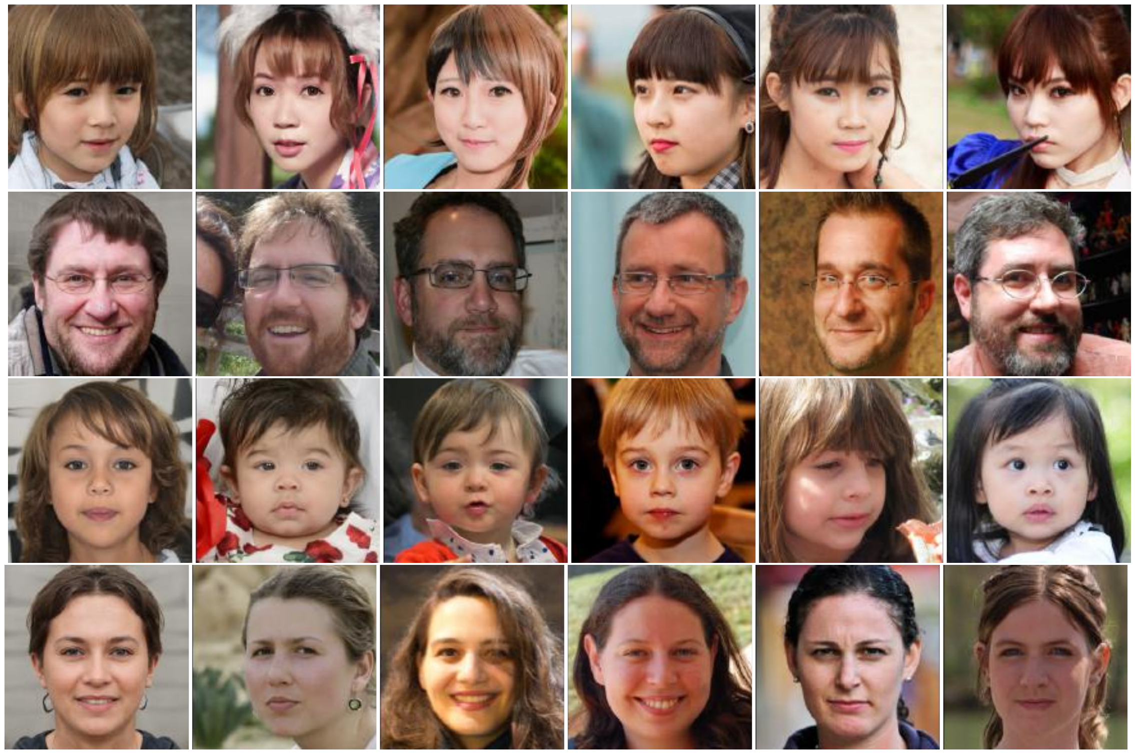


Figure 18: **Nearest neighbors for generated faces**. Within each row, we show a sample from the model on the left. The remaining columns contain real images that are closest to the respective sample in terms of the FaceNet [24] descriptor. The visualization suggests that the CIPS model generalizes well beyond memorization of the training dataset.

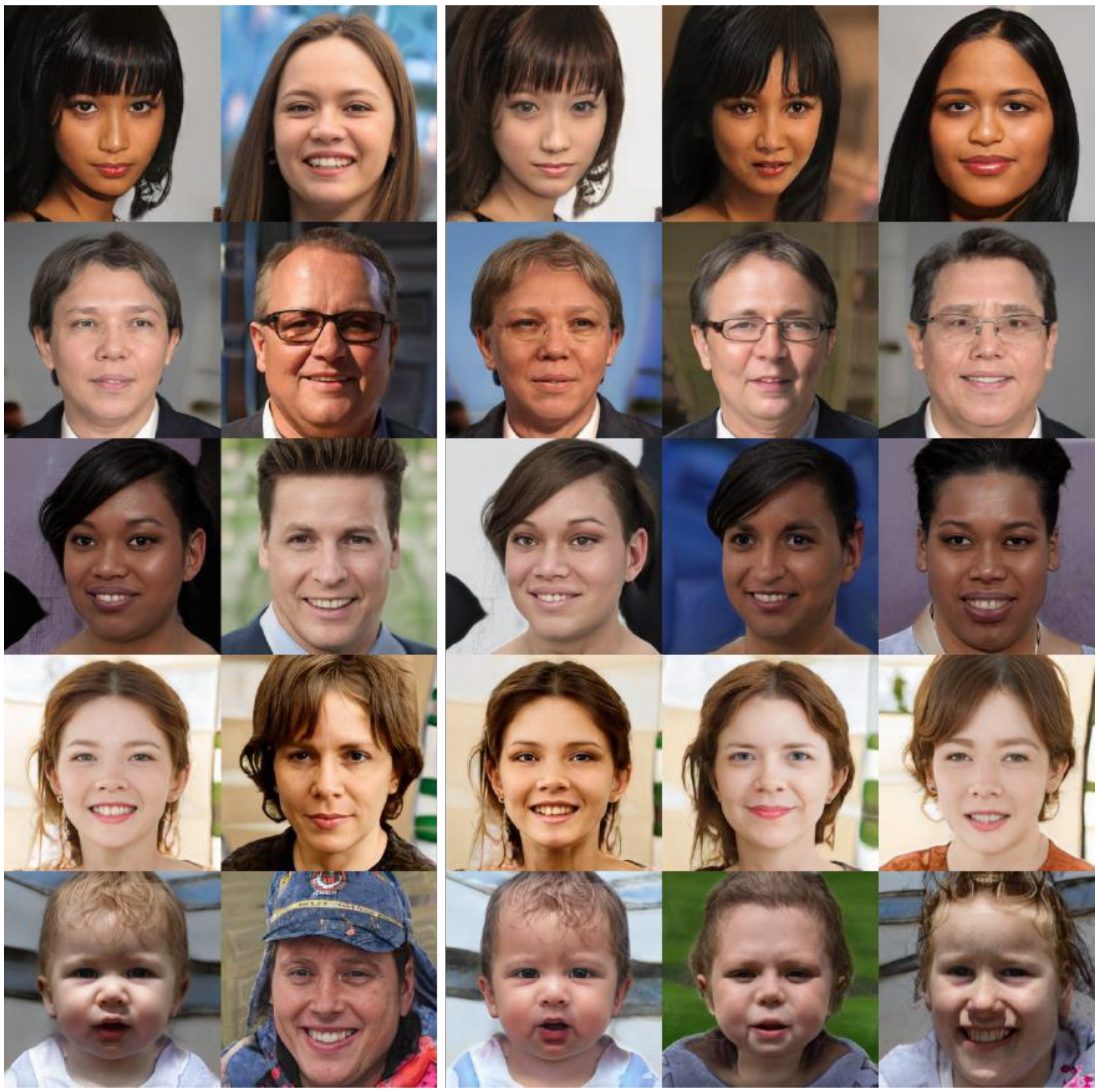


Figure 19: **Layer-wise style mixing**. The two leftmost columns contain source images A and B. In the rightmost three columns, we replace the latent code $\mathbf{w}$ of A with the latent code $\mathbf{w}$ of B at layers (left to right): 6-8, 3-5, 1-2. The visualization suggests that layers 1-2 control the pose and the shape of the head, the middle layers (3-5) control finer geometry such as the shape of eyes, eyebrows and nose, and the final layers (6-8) controls the skin color and the textures. Interestingly, this CIPS model was trained without layerwise mixing, and therefore such decomposition likely arises from the architectural prior.