Interpreting Spatially Infinite Generative Models

Chaochao Lu [1 2] **Richard Turner** [1 3] **Yingzhen Li** [3] **Nate Kushman** [3 4]

Abstract

Traditional deep generative models of images and other spatial modalities can only generate fixed sized outputs. The generated images have exactly the same resolution as the training images, which is dictated by the number of layers in the underlying neural network. Recent work has shown, however, that feeding spatial noise vectors into a fully convolutional neural network enables both generation of arbitrary resolution output images as well as training on arbitrary resolution training images. While this work has provided impressive empirical results, little theoretical interpretation was provided to explain the underlying generative process. In this paper we provide a firm theoretical interpretation for infinite spatial generation, by drawing connections to spatial stochastic processes. We use the resulting intuition to improve upon existing spatially infinite generative models to enable more efficient training through a model that we call an infinite generative adversarial network, or ∞-GAN. Experiments on world map generation, panoramic images and texture synthesis verify the ability of ∞-GAN to efficiently generate images of arbitrary size.

1. Introduction

Generative modeling using neural networks has made significant progress over the last few years. Especially dramatic has been the improvement of models for spatial domains such as images. Deep generative models, such as Generative-Adversarial Networks (GANs) (Goodfellow et al., 2014), have been at the forefront of this effort due to their ability to generate sharp, photo-realistic images (Karras et al., 2018a;b; Brock et al., 2019). Despite this success, traditional generative models can only generate outputs of equal or smaller sizes than the original data on which they were

trained. Furthermore, training on high resolution images is prohibitively expensive, as the size of the network typically must grow at least linearly with the resolution of the training samples. In many domains, however, the resolution of the training samples is either extremely large, or effectively unbounded. Maps and earth imagery such as Landsat images are often very high resolution, and can cover arbitrarily large parts of the planet, resulting in images which are at least millions of pixels in each dimension. Panoramic images can be very large in the horizontal dimension. Texture images for graphics can be arbitrarily large depending on the object to which they are mapped, and medical images can often be $100,000 \times 100,000$ or larger (Komura & Ishikawa, 2018). Recent work (Jetchev et al., 2016; Shaham et al., 2019) has shown that arbitrary sized images can be generated from a fixed sized neural network by feeding in a 2-d grid of noise vectors, i.e. a 3-d tensor of noise, rather than the single vector of noise used in traditional deep generative models. While this work showed impressive empirical results, it did not provide a theoretical basis for these type of models.

In this paper we present a theoretical interpretation of infinite generative modelling based on the idea of *consistent* architectures, where the output generated by a given patch of noise will differ if that patch of noise is generated adjacent to another patch of noise, or on its own. We show that past work has used *inconsistent architectures* which requires removing pixels from the boundary of image patches generated during the incremental generation process required to create large samples. This waste of computation can be significant, and we provide an in-depth analysis of the redundancy in terms of the fraction of discarded pixels. Finally, we prove that by making small changes to the convolutional architectures we can ensure that the generator is a consistent transformation over spatial stochastic processes. This ensures *consistent* outputs, enabling the generated patches to be directly combined without removing pixels.

We use this intuition to design Infinite-GAN (∞-GAN), which we test on texture images, panoramic views and a satellite map of the world. As shown in Figure 1, our model can generate interesting new worlds without simply memorizing the Earth's world map, even when our model only observes 64×64 image patches during training.

---

[1]University of Cambridge, Cambridge, UK [2]Max Planck Institute for Intelligent Systems, Tübingen, Germany [3]Microsoft Research, Cambridge, UK [4]now at DeepMind. Correspondence to: Nate Kushman <nate@kushman.org>.
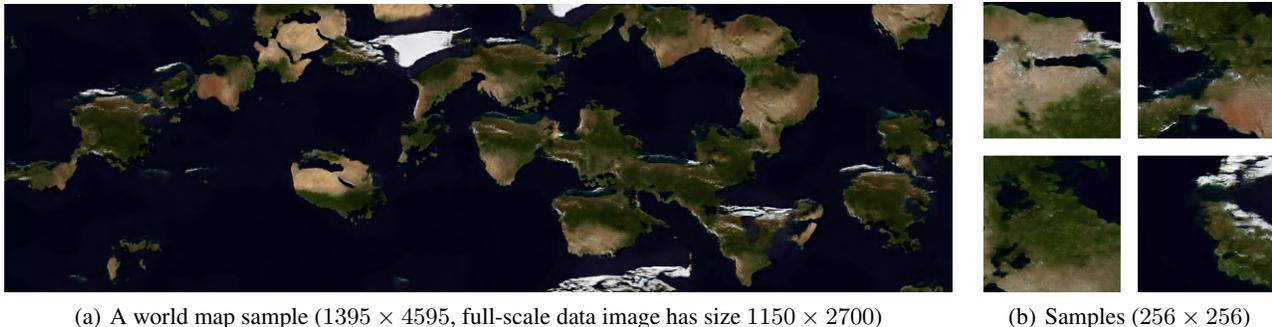
(a) A world map sample (1395×4595, full-scale data image has size 1150×2700) (b) Samples (256×256)

Figure 1. The generated satellite world map from the ∞-GAN (trained on 64×64 patches). We see that the ∞-GAN can generate realistic looking continents with high-fidelity details.

2. Theoretical Results

2.1. Redundancy in incremental generation with inconsistent network architectures

To generate arbitrarily large images with convolutional neural networks (CNNs), the generator $G(\cdot)$ needs to be capable of handling noise inputs of arbitrary size. As convolution and up-samplinng operations are agnostic to the input sizes, a natural idea is to build a Spatial GAN-like generator $G(\cdot)$ with K blocks of {up-sample, conv 3×3 (stride 1, zero padding)} layers, in order to transform a latent tensor $\mathbf{z} \in \mathbb{R}^{h \times w \times C}$ into an image $\mathbf{x} \in \mathbb{R}^{2^K h \times 2^K w \times 3}$. Therefore $G(\cdot)$ is able to generate arbitrary large (but bounded) images *in one shot* by increasing the size of $\mathbf{z}$ to any large (but also bounded) value.

However, *incremental generation* is more suited to generating extremely large images by stitching small patches generated in a sequel. In such case the extension of $G(\cdot)$ to this task is less straight-forward. More importantly, they produce *inconsistent* image patches due to the use of padding. Given $\mathbf{z}_1 \subset \mathbf{z}_2$, the output image $G(\mathbf{z}_1)$ is not a sub-patch of $G(\mathbf{z}_2)$. Similarly when $\mathbf{z}_1 \cap \mathbf{z}_2 \neq \emptyset$, a native stitch of image patches $G(\mathbf{z}_1)$ and $G(\mathbf{z}_2)$ will result in tiling and/or boundary inconsistency artifacts. An ad-hoc solution to this inconsistency issue is to stitch *properly cropped* patches (Jetchev et al., 2016). However, depending on the computational constraints on the maximum spatial size (S, S) for one patch, it can result in discarding many pixels thus wasting a large amount of computation. This redundancy is characterized by the following result proved in Appendix A.

Theorem 1. *Assume the stitched image has spatial size tending towards infinity. Then the fraction of discarded pixels is at least $4(\lfloor \frac{S}{2^K} \rfloor^{-1} - \lfloor \frac{S}{2^K} \rfloor^{-2})$.*

As an example, consider $S = 4096$ and $K = 9$, then the redundancy figure in Theorem 1 is 43.75%. This redundancy can be reduced by decreasing K, but then the network will fail to represent global features in a big image. On the other hand, although increasing K makes the generator more flexible, the benefit is offset by inefficient generation process.

Since one can keep generating patches in the incremental generation process, the output image is effectively unbounded thus *infinite* dimensional. Mathematically this means incremental generation can only be achieved by considering *consistent* transformations over *spatial stochastic processes*; effectively the above incremental generation approach with inconsistent CNNs is one such transformation. Below we establish in theory a *consistent* transform of spatial stochastic processes and discuss a more efficient CNN parameterization of such transformations.

2.2. A convolutional neural network transform for spatial stochastic processes

Consistent transforms for spatial stochastic processes Mathematically, using latent tensor of arbitrary sizes is equivalent to using a spatial stochastic process $\mathbf{z}(\cdot, \cdot) = \{\mathbf{z}(i, j) \in \mathbb{R}^C \mid (i, j) \in \mathbb{Z}^2\}$ as the latent input, e.g. i.i.d. Gaussian noise, i.e. $\mathbf{z}(i, j) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \mathbf{I})$. As computers cannot handle objects of infinite dimensions, in practice we first sample a "noise patch" $\mathbf{z}(J) = \{\mathbf{z}(i, j) \mid (i, j) \in J\}$ with *latent pixel index set* $J = \{1, ..., h\} \times \{1, ..., w\}$ of size (h, w), then pass it through the generator and obtain an image $\mathbf{x}(I) = \{\mathbf{x}(i, j) \in \mathbb{R}^3 \mid (i, j) \in I\} = G(\mathbf{z}(J))$ of size (H, W) with *image pixel index set* $I = \{1, ..., H\} \times \{1, ..., W\}$. For notation ease we write $\{a : b\} = \{a, ..., b\}$ for $a < b$, and when $a = b$, $\{a : b\} = \{a\}$. Also we wish to have the following two desirable properties to allow *incremental* generation of unbounded images from patches:

- Marginalization consistency: for any latent pixel index sets $J' \subset J$ of finite sizes (and the corresponding image pixel index sets $I' \subset I$), taking the patch with index set I' from $\mathbf{x}(I) = G(\mathbf{z}(J))$ is equivalent to directly generating $\mathbf{x}(I') = G(\mathbf{z}(J'))$;

- Permutation invariance: generating two patches $\mathbf{x}(I), \mathbf{x}(I'')$ in the same image is invariant to the order of generation.

If these two conditions are meet, then incremental image generation can be achieved by cutting the latent pixel index

Table 1. Summary of consistent and stationarity preserving transforms on (stationary) spatial processes. *Proved for commonly used activation functions (sigmoid, ReLU, etc.). **It transforms a strictly stationary spatial process to a cyclostationary spatial process.

layer	consistent?	stationarity preserving?
convolution (zero/constant padding)	No	No (not a valid transform)
convolution (no padding, stride 1)	Yes	Yes
pixel-wise non-linearity	Yes*	Yes
up-sampling (nearest, scale 2)	Yes	Yes**
up-sampling (bi-linear, scale 2, boundary crop)	Yes	Yes**
pixel normalization	Yes	Yes

Figure 2. Visualizing cyclostationarity with period $(2, 1)$.

set J into subsets $\{J^n \mid \cup_n J^n = J, J^{n_1} \cap J^{n_2} \neq \emptyset\}$ with proper index subset overlap, sampling $\boldsymbol{z}(J)$. and stitching the individual patches $G(\boldsymbol{z}(J^n))$ together (without cropping) to obtain $\boldsymbol{x}(I) = \cup_n G(\boldsymbol{z}(J^n))$. Marginalization consistency ensures that $G(\boldsymbol{z}(J)) = \cup_n G(\boldsymbol{z}(J^n))$, and permutation invariance allows each patch $G(\boldsymbol{z}(J^n))$ to be generated in random order (provided an algorithm to retrieve the corresponding subset sample $\boldsymbol{z}(J^n)$ from $\boldsymbol{z}(J)$).

The two properties can be ensured by the *consistency* requirement for $G(\cdot)$: denote the corresponding operator on stochastic processes as $\mathcal{O}_G$, then the output $\boldsymbol{x}(\cdot, \cdot) = \{\boldsymbol{x}(i, j) \in \mathbb{R}^3 \mid (i, j) \in \mathbb{Z}^2\} = \mathcal{O}_G(\boldsymbol{z}(\cdot, \cdot))$ also needs to be a spatial stochastic process. We prove in Appendix B.1 the consistency results for popular CNN building blocks summarized in Table 1. Specifically, padding is discarded to remove inconsistency at the boundary of each layer. Similarly the bi-linear up-sampling operation (scale 2) is adapted by cropping out the boundary pixels with edge width 1. Our generative network (discussed below) only uses consistent transforms, and we have the following result.

Theorem 2. $\mathcal{O}_G$ *is a consistent operator on spatial stochastic processes if $G(\cdot)$ is constructed using the consistent transforms presented in Table 1.*

The stationarity pattern from CNN transforms A generative model can only see finite number of images of finite size in training, thus without further assumptions, the generator cannot generate realistic looking images of sizes that are larger than the maximum size of observed images.

However, if the input $\boldsymbol{z}(\cdot, \cdot)$ in training time is a *strictly stationary* spatial stochastic process, e.g. i.i.d. Gaussians, then for any index set $J \subset \mathbb{Z}^2$, the distribution of $\boldsymbol{z}(J)$ is shift-invariant for any shifting direction. Thus if the generator $G(\cdot)$ preserves stationarity on patch level, then during training it suffice to fix e.g. $J = \{1 : h\} \times \{1 : w\}$, and match the distribution of $\boldsymbol{x}(I) = G(\boldsymbol{z}(J))$ to the distribution of training image patches of size (H, W).

In math, this requires the generator to be able to transform a strictly stationary process into a cyclostationary process.

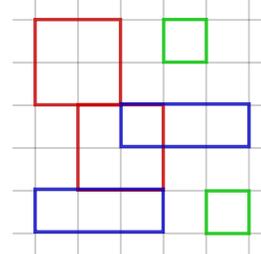**Definition 1.** *A spatial stochastic process $\boldsymbol{x}(\cdot, \cdot)$ is cyclo-* *stationary with period (H, W) if for all $I = \{a : b\} \times \{c : d\} \subset \mathbb{Z}^2$ and $\tilde{I} = \{a + H : b + H\} \times \{c + W : d + W\}$, $\boldsymbol{x}(I) \overset{d}{=} \boldsymbol{x}(\tilde{I})$.*

We visualize cyclostationarity with period $(2, 1)$ in Figure 2; in this figure patches with bounding boxes in the same color have the same distribution. Note that strictly stationary is cyclostationary with period $(1, 1)$; in general the pixels inside an (H, W) patch do not necessarily have the same marginal distributions. In Appendix B.2 we investigate the *stationarity preserving* properties, i.e. given a cyclostationary input process, whether the output process is still cyclostationary, for CNN building blocks. Results are again summarized in Table 1; using stationarity preserving transforms as CNN layers, we can prove the following result.

Theorem 3. $\mathcal{O}_G$ *preserves cyclostationarity if $G(\cdot)$ is constructed using the stationarity preserving transforms presented in Table 1.*

We note that using up-sampling of scale 2 increases the stationarity period by 2. For i.i.d. Gaussian inputs, this means $\boldsymbol{x}(\cdot, \cdot)$ has stationarity period $(2^K, 2^K)$ if $G(\cdot)$ uses K up-sampling layers.

3. Experiments

To further demonstrate the theoretical results above, we use the theory to design ∞-GAN, which is built on Spatial GAN (Jetchev et al., 2016) and SinGAN (Shaham et al., 2019). There are two major modifications: (1) ∞-GAN's generators are constructed only using the consistent and stationarity preserving transforms; (2) we slightly modify SinGAN's multi-scale architecture to improve computational efficiency by training with a constant image patch size regardless of the output resolution. Details of the ∞-GAN architecture are illustrated in Appendix D.

We evaluate the ∞-GAN model on three datasets of large images: a satellite map of the world, a panoramic image, and texture images. We present the map generation results in the main text and refer the readers to Appendix E for others. Details on data collection, network architectures and hyper-parameter tuning is presented in Appendix F.

Table 2. **Satellite image FID scores** (mean $\pm$ std.) from a bootstrap estimate of FID (5 random crops). The ∞-GAN with the proposed multi-scale training significantly outperforms the baselines.

model / patch size	32×32	64×64	128×128	256×256
Spatial GAN	40.93 ± 0.68	43.75 ± 0.33	54.41 ± 0.25	71.69 ± 0.25
PSGAN	34.55 ± 0.25	48.77 ± 0.20	64.93 ± 0.55	81.65 ± 0.32
∞-GAN - $\mathcal{D}_2$ Only	39.87 ± 0.35	67.05 ± 0.37	95.12 ± 0.91	133.75 ± 0.27
∞-GAN - Full	12.09 ± 0.25	24.18 ± 0.09	45.35 ± 0.22	56.60 ± 0.31

(a) Original Data (b) Spatial GAN (c) PSGAN sample (d) $\mathcal{D}_2$ only model (e) ∞-GAN sample

(f) Data patches (g) Spatial GAN (h) PSGAN patches (i) $\mathcal{D}_2$ only model (j) ∞-GAN
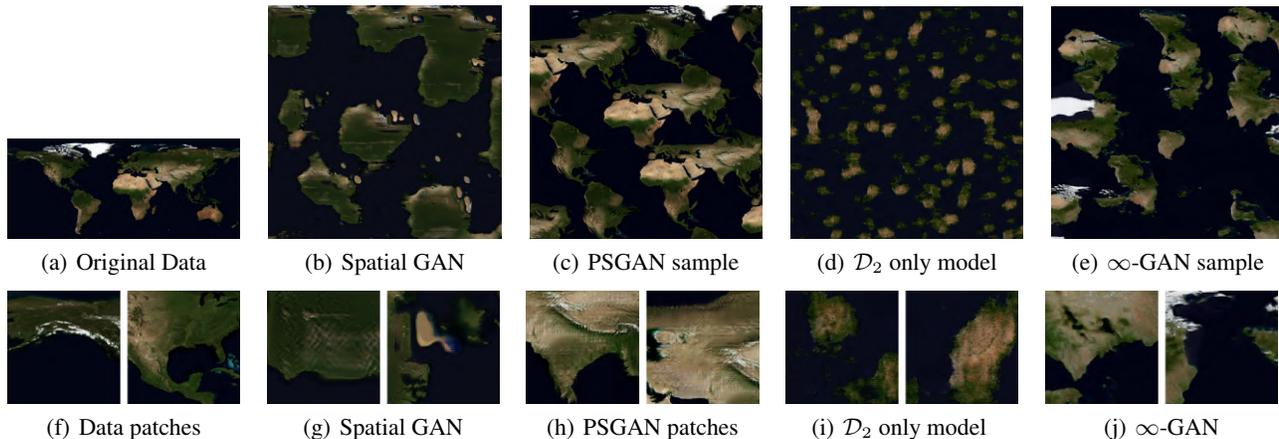
Figure 3. World map samples. The top row shows samples of size 2048×2048 (except for the data), and the bottom rows shows data/generated patches of size 256×256.

Generated samples up to size 4096x4096 are provided in this url.

3.1. Evaluation on Satellite World Map

We consider the task of generating novel and realistic looking satellite world maps. Our ∞-GAN model is trained on a 1150×2700 satellite image taken from the NASA visible Earth project (see Appendix F). Training patches are obtained by first down-sizing the modified world map image by $1, 2$ and $4\times$, then randomly cropping patches of size 64×64. This returns three datasets $\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2$ containing images patches with increasing resolutions, which are used to train the multi-scale generator at different scale levels.

Here we consider three baseline models for comparison. The first baseline is a Spatial GAN, and the second basesline is a Periodic Spatial GAN (PSGAN) (Bergmann et al., 2017) which is a Spatial GAN with global latent variables and learnable periodic embeddings. Both models has model patch size 256×256 and they are trained on 256×256 crops from the full-scale image. The third baseline is our extension network $G_0(\cdot)$ trained on 64×64 patches from $\mathcal{D}_2$, which represents a version of our model without the multi-scale architecture. We refer to this baseline as the $\mathcal{D}_2$ only model. We compare the models quantitatively using the Frechet Inception Distance (FID) (Heusel et al., 2017) on the generated outputs with $\mathcal{D}_2$ level resolution, by

generating 5 images of size 2048×2048, sampling 10,000 random crops to construct a set of "fake image patches", and then comparing with the random cropped patches from the full-scale data image. The patch-level FID is computed using different crop sizes; with small patch size the score measures the quality in terms of terrain texture, and with large patch size the score evaluates the high-level structures of the generated images.

The FID scores are reported in Table 2. We see that both the Spatial GAN and the PSGAN fail to capture the terrain textures from the real world map. They perform slightly better than the $\mathcal{D}_2$ only model in terms of FID scores on larger patches, which is reasonable as the $\mathcal{D}_2$ only model is trained on 64×64 patches only, therefore the $\mathcal{D}_2$ only model cannot represent the high-level structures in the original world map. On the other hand, the full ∞-GAN with fixed size multi-scale training significantly outperforms the baselines at all patch levels.

The quantitative results are supported by the visualizations in Figure 3. Here the $\mathcal{D}_2$ only model fails to capture global statistics and so it only generates small islands and never generates a continent. Although the spatial GAN can generate continents, the image quality is much lower than our model. PSGAN, on the other hand, fails to generate diverse outputs, showing clear issues of data memorization. Compared to the baselines, the ∞-GAN model can generate

novel world maps without just memorizing the data. The generated map contains realistic looking continents, and the image quality is significantly better than the two baselines.

4. Discussion

We presented a theory of infinity in generative modelling, based on which deep generative models are capable of generating unbounded images of high resolutions. According to our theory, we provided an example of ∞-GAN, whose generator is built with a consistent transform over spatial stochastic processes. This design is crucial for efficient incremental generation compared to inconsistent generators such as the Spatial GAN. The model is trained by a novel multi-scale learning algorithm using images of fixed size, which does not require full-scale high resolution images for learning. Experiments showed that the ∞-GAN can capture both the global dependencies shared across patches, as well as the details for each of the high resolution patches, which further demonstrated our theory.

Our work is related to research in generating high fidelity images (Karras et al., 2018a;b; Brock et al., 2019), stochastic processes (see e.g. Huang, 1984; Haining, 1978; Ripley; Anselin, 2013), super-resolution (Ledig et al., 2017; Sønderby et al., 2016; Wang et al., 2018) and texture synthesis (Bergmann et al., 2017; Jetchev et al., 2016). These related works are discussed extensively in Appendix C.

The closest related work in the context of application is Sin-GAN (Shaham et al., 2019) which, after trained on a single image, can also generate images of arbitrary size at test time by changing the dimensions of the noise maps. They also conducted an empirical investigation on the artifacts introduced by padding in terms of reduced diversity, and their solution followed the CNN architecture of Ioffe & Szegedy (2015) and remove padding in the convolutional layer. Our theoretical analysis goes one step further to analyse popular architectural choices in building convolutional generators, and the usage of consistent and stationarity preserving transforms ensures the validity of ∞-GAN's generative model. Another difference is that the generators in SinGAN take as input the images of increasing size as training progresses. By contrast, we feed image patches of fixed size at all resolutions into ∞-GAN's generators in the hierarchy, which enables more efficient training.

Future work will consider extending our theory to the non-stationary case. Advanced neural network models will be developed to represent non-stationary information in a spatial stochastic process.

References

Anselin, L. *Spatial econometrics: methods and models*, volume 4. Springer Science & Business Media, 2013.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, volume 28, pp. 24. ACM, 2009.

Beckham, C. and Pal, C. A step towards procedural terrain generation with gans. *arXiv preprint arXiv:1707.03383*, 2017.

Bergmann, U., Jetchev, N., and Vollgraf, R. Learning texture manifolds with the periodic spatial gan. In *International Conference on Machine Learning*, pp. 469–477, 2017.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1xsqj09Fm`.

Denton, E. L., Chintala, S., Fergus, R., et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pp. 1486–1494, 2015.

Efros, A. A. and Freeman, W. T. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 341–346. ACM, 2001.

Efros, A. A. and Leung, T. K. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pp. 1033–1038. IEEE, 1999.

Fournier, A., Fussell, D., and Carpenter, L. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, 1982.

Gatys, L., Ecker, A. S., and Bethge, M. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pp. 262–270, 2015a.

Gatys, L. A., Ecker, A. S., and Bethge, M. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015b.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Haining, R. The moving average model for spatial interaction. *Transactions of the Institute of British Geographers*, pp. 202–225, 1978.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Huang, H., He, R., Sun, Z., Tan, T., et al. Introvae: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing Systems*, pp. 52–63, 2018.

Huang, J. S. The autoregressive moving average model for spatial analysis. *Australian Journal of Statistics*, 26(2): 169–178, 1984.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Jetchev, N., Bergmann, U., and Vollgraf, R. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=Hk99zCeAb.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018b.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Komura, D. and Ishikawa, S. Machine learning methods for histopathological image analysis. *Computational and structural biotechnology journal*, 16:34–42, 2018.

Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (ToG)*, 22(3):277–286, 2003.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

Li, C. and Wand, M. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pp. 702–716. Springer, 2016.

Ma, C., Li, Y., and Hernández-Lobato, J. M. Variational implicit processes. *arXiv preprint arXiv:1806.02390*, 2018.

Menick, J. and Kalchbrenner, N. GENERATING HIGH FIDELITY IMAGES WITH SUBSCALE PIXEL NETWORKS AND MULTIDIMENSIONAL UPSCALING. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HylzTiC5Km.

Musgrave, F. K., Kolb, C. E., and Mace, R. S. The synthesis and rendering of eroded fractal terrains. In *ACM Siggraph Computer Graphics*, volume 23, pp. 41–50. ACM, 1989.

Olsen, J. Realtime procedural terrain generation. *Technical Report, University of Southern Denmark*, 2004.

Perlin, K. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, July 1985. ISSN 0097-8930. doi: 10.1145/325165.325247. URL http://doi.acm.org/10.1145/325165.325247.

Perlin, K. Improving noise. In *ACM transactions on graphics (TOG)*, volume 21, pp. 681–682. ACM, 2002.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Reed, S., Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Chen, Y., Belov, D., and Freitas, N. Parallel multiscale autoregressive density estimation. In *International Conference on Machine Learning*, pp. 2912–2921, 2017.

Ripley, B. D. *Spatial statistics*, volume 24. Wiley Online Library.

Robinson, P. The estimation of a nonlinear moving average model. *Stochastic processes and their applications*, 5(1): 81–90, 1977.

Shaham, T. R., Dekel, T., and Michaeli, T. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4570–4580, 2019.

Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

Ulyanov, D., Lebedev, V., Lempitsky, V., et al. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning*, pp. 1349–1357, 2016a.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016b.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6924–6932, 2017.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8798–8807, 2018.

Wei, L.-Y. and Levoy, M. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5907–5915, 2017.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

A. Redundancy of incremental generation with inconsistent architectures

We assume an inconsistent generator $G(\cdot)$ contains K blocks of {nearest up-sampling, conv 3×3} layers. Here the up-sampling scale is $U = 2$ and the 3×3 convolution uses stride 1 and zero padding. Therefore given z input of spatial size (N, N), the output spatial size is $(2^K N, 2^K N)$. Assume the computational budget allows generations of patches of size at most $S \times S$. This means K is at most $\lfloor \log_2 S \rfloor$.

We define *inconsistent pixels* in an image patch as the pixels that are dependent on at least one of the zeros padded in the convolutional layers. Other pixels in that patch are *consistently generated*, and in incremental generation, we only maintain consistent pixels and throw away the inconsistent ones.

Recall the notation for the input index set as $J = \{a : b-1\}^2$ with $\{a : b-1\} = \{a, a+1, ..., b-1\}$. For convenience in the rest of this section we also write $[a : b) = \{a : b-1\} = \{a, a+1, ..., b-1\}$ using python's indexing method.

Lemma A.1. *If the input index set is $J = [a : b)^2$, then the output of the generator $x(I^K) = G(z(J))$ has index set $I^K = [2^K a : 2^K b)^2$, and only the pixels in $x(\tilde{I}^K)$ with $\tilde{I}^K = [2^K a + 2^K - 1 : 2^K b - 2^K + 1)^2$ are consistently generated.*

Proof. It is clear that a nearest up-sampling layer with scale 2 maps a pixel with index (i, j) to a 2×2 patch with index set $\{2i : 2i+1\} \times \{2j : 2j+1\}$. Also notice that convolution with zero padding does not change the spatial dimension. Therefore after K blocks of transforms, the upper-left corner pixel in latent space $z(a, a)$ gets mapped to the upper-left corner of the output image patch with the upper-left corner pixel $x(2^K a, 2^K a)$. At the same time, the bottom-right corner pixel in latent space $z(b-1, b-1)$ gets mapped to the bottom-right corner of the output image patch with the bottom-right corner pixel $x(2^K b - 1, 2^K b - 1)$. Therefore the output index set is $I^K = [2^K a : 2^K b)^2$.

We prove the second part of the lemma by induction. Assume at the $(k-1)^{th}$ block the output patch has index set $[\alpha : \beta)^2$, and it has inconsistent feature variables around the edge with width w. This means after the kth up-sampling, the set of inconsistent feature variables are also around the edge with width $2w$. Then after the 3×3 convolution, the output feature variables with index $i \in \{2\alpha + 2w + 1, 2\beta - 2w - 1\}$ and/or $j \in \{2\alpha + 2w + 1, 2\beta - 2w - 1\}$ are dependant on the inconsistent feature variables in the last layer thereby inconsistent as well. Therefore at the kth block the output patch has index set $[2\alpha : 2\beta)^2$, and the inconsistent feature variables are around the edge with width $2w + 1$. Since the input of the 1st block are all consistent latent variables, the image ouput $x(I^K)$ has $2^K - 1$ inconsistent pixels along each side of the image. Therefore the consistent pixels are $x(\tilde{I}^K)$ with $\tilde{I}^K = [2^K a + 2^K - 1 : 2^K b - 2^K + 1)^2$. $\square$

In incremental generation of neighbouring patches $x(I_1^K)$ and $x(I_2^K)$, the corresponding index sets J_1 and J_2 in z need to have overlap. Otherwise the patches are independent to each other, which is undesirable. However, if the overlap $J_1 \cap J_2$ is too large, then many consistent pixels in the image space get wasted. So in the following lemma we calculate the overlap size which achieves the lowest redundancy of generated pixels.

Lemma A.2. *To achieve consistent incremental generation of two neighbouring patches, J_1 and J_2 only needs to overlap by 2 columns or 2 rows in z space, when $K \geq 2$ blocks are in use. In this case the consistent output patches $x(\tilde{I}_1^K)$ and $x(\tilde{I}_2^K)$ have overlapping pixels of 2 columns or 2 rows.*

Proof. We prove this overlap result for row overlapping, column overlapping result can be proved accordingly. We assume $J_1 = [a_1 : b_1) \times [\alpha : \beta)$ and $J_1 = [a_2 : b_2) \times [\alpha : \beta)$. Assume $a_1 < a_2 < b_1 < b_2$, it is equivalent to prove $b_1 - a_2 \geq 2$ (as now the overlapping rows in z space has row indices $\{a_2, a_2 + 1, ...b_1 - 1\}$).

From Lemma A.1 we see that the output patches $x(I_1^K)$ and $x(I_2^K)$ has index sets $I_1^K = [2^K a_1 : 2^K b_1) \times [2^K \alpha : 2^K \beta)$ and $I_2^K = [2^K a_2 : 2^K b_2) \times [2^K \alpha : 2^K \beta)$. However since the inconsistent pixels need to be removed, the resulting consistent patches has index sets $\tilde{I}_1^K = [2^K a_1 + 2^K - 1 : 2^K b_1 - 2^K + 1) \times [2^K \alpha + 2^K - 1 : 2^K \beta - 2^K + 1)$ and $\tilde{I}_2^K = [2^K a_2 + 2^K - 1 : 2^K b_2 - 2^K + 1) \times [2^K \alpha + 2^K - 1 : 2^K \beta - 2^K + 1)$. To make the two consistent patches $x(\tilde{I}_1^K)$ and $x(\tilde{I}_2^K)$ overlap or at least adjacent, it requires $2^K b_1 - 2^K + 1 \geq 2^K a_2 + 2^K - 1$, which means $b_1 \geq a_2 + 2 - 1/2^{K-1}$. As $K \geq 1$ and $b_1, a_2 \in \mathbb{Z}$, this means $b_1 - a_2 \geq 2$ when $K \geq 2$ and $b_1 - a_2 \geq 1$ when $K = 1$.

To prove the second part of the lemma, we set $a_2 = b_1 - 2$. We also assume $b_1 - a_1 = b_2 - a_2 = \alpha - \beta = \lfloor \log_2 S \rfloor$ in order to minimize the number of inconsistent pixels to be discarded. This means the consistent output patches $x(\tilde{I}_1^K)$ and $x(\tilde{I}_2^K)$ have overlapping pixels, i.e. $\tilde{I}_1^K \cap \tilde{I}_2^K = [2^K a_2 + 2^K - 1 : 2^K b_1 - 2^K + 1) \times [2^K \alpha + 2^K - 1 : 2^K \beta - 2^K + 1)$. Note that $2^K b_1 - 2^K + 1 - (2^K a_2 + 2^K - 1) = 2$ since $b_1 - a_2 = 2$. Therefore the two consistent output patches have overlapping pixels of 2 rows. $\square$

Lemma A.2 indicates that the two rows of consistent pixels $\boldsymbol{x}([2^K a_2 + 2^K - 1 : 2^K b_1 - 2^K + 1) \times [2^K \alpha + 2^K - 1 : 2^K \beta - 2^K + 1))$ is generated twice, and if $\boldsymbol{x}(I_2^K)$ is generated after $\boldsymbol{x}(I_1^K)$, then we need to discard these two rows of pixels in $\boldsymbol{x}(\tilde{I}_2^K)$. So in sum, for an output patch $\boldsymbol{x}(I^K)$ that is one of the interior patches in the final huge image (which is stitched from $M \times M$ patches), $2(2^K - 1)$ rows/columns of inconsistent pixels as well as 2 rows/columns of consistent pixels need to be discarded. As the final stitched image has $4M - 4$ boundary patches but $(M - 2)^2$ interior patches, it means when $M \to +\infty$, the redundancy of pixel generation is dominated by the redundancy in generating the interior patches. Therefore we can prove Theorem 1 presented in the main text.

Proof. As explained above it is sufficient to compute the fraction of discarded pixels in an interior patch. Recall the input $\boldsymbol{z}(J)$ has spatial size (N, N). Then in $\boldsymbol{x}(I^K)$ which has spatial size $(2^K N, 2^K N)$, 2^{K+1} rows and 2^{K+1} columns of pixels are discarded. Therefore the percentage of discarded pixels is

$$1 - \frac{(2^K N - 2^{K+1})^2}{2^{2K} N^2} = 1 - \frac{(N - 2)^2}{N^2} = \frac{4}{N} - \frac{4}{N^2}.$$

Now recall that $2^K N \leq S$, meaning that $N \leq \lfloor \frac{S}{2^K} \rfloor$. This means the pixel generation redundancy is of percentage at least $4[(\lfloor \frac{S}{2^K} \rfloor)^{-1} - (\lfloor \frac{S}{2^K} \rfloor)^{-2}]$. $\square$

One can reduce the number of redundantly generated pixels by decreasing K, the number of {nearest up-sampling, conv 3×3} blocks. However when K is small the network will fail to capture the global information presented in a big image patch. On the other hand, with large K, although the generator gets very flexible, in incremental generation this flexibility is severely damaged as a large fraction of pixels in patches are removed. As the overlap in $\boldsymbol{z}$ space is 2, this also means $N \geq 3$ and the number of blocks K is restricted to have $K \leq \lceil \log_2 \frac{S}{3} \rceil$. Even in this case the fraction of removed pixels, when $M \to +\infty$, goes to $8/9$ which is a very significant value.

B. Proofs for consistency and stationarity preserving results

B.1. Proofs for consistency

We wish to establish the consistency results for operators defined by a convolutional neural network. It it sufficient to prove that each component of this convolutional neural network, including convolution, pixel-wise non-linearity and up-sampling (bilinear or nearest interpolation), is a valid operator on stochastic processes, i.e. the output of the operation is also a stochastic process. We follow Ma et al. (2018) to construct our proofs, before that we explain in below the proof ideas of the consistency results. The main techniques are the Kolmogorov extension theorem and the Karhunen-Loeve expansion, and we use either of which is more convenient than the other.

We assume w.l.o.g. the input stochastic process as $\boldsymbol{z}(\cdot, \cdot) \sim SP(0, \mathcal{K})$ a centered discrete stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$, i.e. $C = 1$ and $\boldsymbol{z}(i, j) \in \mathbb{R}$. Define the collection of random variables $\boldsymbol{x}(\cdot, \cdot) = \{\boldsymbol{x}(i, j) = \mathcal{O}(\boldsymbol{z}(\cdot, \cdot))(i, j) \mid (i, j) \in \mathbb{Z}^2\}$ as the outcome of a given operator $\mathcal{O}$. We say the operator $\mathcal{O}$ is *consistent* if $\boldsymbol{x}(\cdot, \cdot)$ is also a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$.

Also denote the distribution of a finite subset $\boldsymbol{x}(I) = \{\boldsymbol{x}(i, j) \mid (i, j) \in I\} \subset \boldsymbol{x}(\cdot, \cdot)$ with index set $I \subset \mathbb{Z}^2$ as $P_{X,I}(\boldsymbol{s})$. The Kolmogorov extension theorem states the consistency conditions of a stochastic process as:

1. Marginalization consistency: for any finite subset $I \subset \mathbb{Z}^2$, and any $I' \subset I$,

$$P_{X,I'}(\boldsymbol{x}(I')) = P_{X,I}(\boldsymbol{x}(I')) := \int P_{X,I}(\boldsymbol{x}(I)) d\boldsymbol{x}(I \backslash I').$$

2. Permutation invance: for any permutation I'' of I,

$$P_{X,I''}(\boldsymbol{x}(I'')) = P_{X,I}(\boldsymbol{x}(I)).$$

Therefore given an operator $\mathcal{O}$, one can check the two conditions for $\boldsymbol{x}(\cdot, \cdot)$ to validate whether the output is a stochastic process. When $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process, we may omit the subscript I and write the distribution as $P_X(\boldsymbol{x}(I))$.

One can also use the he Karhunen-Loeve expansion (K-L expansion) theorem to prove the existence of a stochastic process. This input stochastic process can be K-L expanded as a stochastic infinite series

$$\boldsymbol{z}(i, j) = \sum_{m}^{\infty} \sqrt{\lambda_m} Z_m \phi_m(i, j), \quad \sum_{m}^{\infty} \lambda_m < +\infty, \tag{1}$$

where $\{Z_m\}$ is a collection of zero-mean, unit-variance, uncorrelated variables, and $\{\phi_m(\cdot, \cdot)\}$ is an orthonomal basis of $\mathcal{L}^2(\mathbb{Z}^2)$. Our proofs use the K-L expansion of $\boldsymbol{z}(\cdot, \cdot)$ and establish the convergence of the operator applied on the stochastic infinite series in $\mathcal{L}^2(\mathbb{Z}^2)$. Since in this case $\boldsymbol{x}(\cdot, \cdot)$ can also be represented as a convergent stochastic infinite series, it means that $\boldsymbol{x}(\cdot, \cdot)$ is also a stochastic process.

In the following lemmas we consider the operator as convolution, pixel-wise non-linear transformation and bi-linear up-sampling.

Lemma B.1. *The convolution operator with finite filter size K, stride 1 and **zero/constant padding** is inconsistent.*

Proof. The convolution operator $\mathcal{O}$ transforms a $H \times W \times C$ image patch tensor to another $H \times W \times C$ image patch by padding in zeros or constants around the edge of the input patch. This violates the marginalization consistency requirement in Kolmogorov extension theorem. By taking $I = \{1 : H + 1\} \times \{1 : W + 1\}$ and $I' = \{1 : H\} \times \{1 : W\}$ and using the input process $\boldsymbol{z}(\cdot, \cdot)$ as i.i.d. Gaussian with non-zero mean and marginal variance $\sigma^2 \to 0$, it is straight-forward to see the equality in the marginalization consistency requirement does not hold. $\square$

Lemma B.2. *For a convolution operator with finite filter size K, stride 1 and **no padding**, $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$ if $\boldsymbol{z}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$.*

Proof. Here we consider w.l.o.g. a convolutional filter with filter size $K = 2K' + 1$ output channel 1. This means the parameters of the convolutional filter can be represented as $F \in \mathbb{R}^{K \times K \times C}$, $F = \{F(k, k') \in \mathbb{R}^C \mid (k, k') \in \{-K' : K'\}^2\}$ and the convolution operation is defined as

$$\boldsymbol{x}(i, j) = \mathcal{O}(\boldsymbol{z}(\cdot, \cdot))(i, j) = \sum_{k=-K'}^{K'} \sum_{k'=-K'}^{K'} \langle F(k, k'), \boldsymbol{z}(i - k, j - k') \rangle.$$

It is straight-forward to show that

$$\tilde{\boldsymbol{x}}(i, j) = \sum_{k=-K'}^{K'} \sum_{k'=-K'}^{K'} |\langle F(k, k'), \boldsymbol{z}(i - k, j - k') \rangle| < \infty$$

as $F(k, k') \in \mathbb{R}^C$, $\boldsymbol{z}(i, j) \in \mathbb{R}^C$, a dot product in $\mathbb{R}^C$ is finite, and the above equation is a finite sum. Therefore we can show via Fubini's theorem, $|\boldsymbol{x}(i, j)| \le \tilde{\boldsymbol{x}}(i, j) < +\infty$, and

$$\begin{aligned}
\boldsymbol{x}(i, j) &= \sum_{k=-K'}^{K'} \sum_{k'=-K'}^{K'} \langle F(k, k'), \sum_{m}^{\infty} \sqrt{\lambda_m} Z_m \phi_m(i - k, j - k') \rangle \\
&= \sum_{m}^{\infty} \sqrt{\lambda_m} Z_m \sum_{k=-K'}^{K'} \sum_{k'=-K'}^{K'} \langle F(k, k'), \phi_m(i - k, j - k') \rangle.
\end{aligned} \tag{2}$$

To show this stochastic infinite series converge in $\mathcal{L}^2(\mathbb{Z}^2)$, we need to show

$$||\boldsymbol{x}(\cdot, \cdot)||_{\mathcal{L}^2(\mathbb{Z}^2)}^2 \le ||\mathcal{O}||^2 || \sum_{m}^{\infty} \sqrt{\lambda_m} Z_m \phi_m(\cdot, \cdot) ||_{\mathcal{L}^2(\mathbb{Z}^2)}^2 = ||\mathcal{O}||^2 || \sum_{m}^{\infty} \lambda_m ||Z_m||_2^2 < +\infty.$$

Here the operator norm is defined as

$$||\mathcal{O}|| := \inf\{a \ge 0 \mid ||\mathcal{O}(f)||_{\mathcal{L}^2(\mathbb{Z}^2)} \le a ||f||_{\mathcal{L}^2(\mathbb{Z}^2)}, \forall f \in \mathcal{L}^2(\mathbb{Z}^2)\},$$

and it is straight-forward to show for convolution, $||\mathcal{O}|| = \max |F(k, k', c)|$, which is the the maximum absolute value of the filter tensor F thus finite. Also $\sum_m^{\infty} \lambda_m ||Z_m||_2^2$ converges almost surely since $\sum_m^{\infty} \lambda_m < +\infty$. Therefore $||\boldsymbol{x}(\cdot, \cdot)||_{\mathcal{L}^2(\mathbb{Z}^2)}^2 < +\infty$, and $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$. $\square$

Lemma B.3. *For a pixel-wise nonlinear operator defined by an activation function $f(\cdot)$, if there exists $0 \leq A < +\infty$ such that $f(x) \leq A|x|, \forall x \in \mathbb{R}$, then $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$ if $\boldsymbol{z}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$.*

Proof. The pixel-wise nonlinear operator with activation function $f(\cdot)$ is defined as

$$\boldsymbol{x}(i,j) = \mathcal{O}(\boldsymbol{z}(\cdot, \cdot))(i,j) = f(\boldsymbol{z}(i,j)).$$

By the assumption on $f(\cdot)$ we have $|\boldsymbol{x}(i,j)| \leq A|\boldsymbol{z}(i,j)|$. Since $||\boldsymbol{z}(\cdot, \cdot)||^2_{\mathcal{L}^2(\mathbb{Z}^2)} < +\infty$, this also indicates that $||\boldsymbol{x}(\cdot, \cdot)||^2_{\mathcal{L}^2(\mathbb{Z}^2)} < +\infty$, and $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$. □

We note that commonly used activation functions (sigmoid, ReLU, etc.) satisfy the condition required by this lemma. This condition is also not a necessary condition, see the proof for the following lemma which can also be extended to commonly used activation functions.

Lemma B.4. *Pixel normalization is a consistent transform for stochastic processes.*

Proof. Assume $\boldsymbol{z}(i,j) \in \mathbb{R}^C$ and each of the values in a channel is $\boldsymbol{z}(i,j,c)$. Pixel normalization is defined as the following:

$$\boldsymbol{x}(i,j) = \mathrm{PixelNorm}(\boldsymbol{z}(i,j)) = \frac{\boldsymbol{z}(i,j) - \mu(i,j)}{\sigma(i,j)},$$

$$\mu(i,j) = \frac{1}{C} \sum_{c=1}^{C} \boldsymbol{z}(i,j,c), \quad \sigma^2(i,j) = \frac{1}{C} \sum_{c=1}^{C} (\boldsymbol{z}(i,j,c) - \mu(i,j))^2.$$

Recall that the permutation test is performed on pixel level. As PixelNorm is performed on pixel level as well, it means the permutation invariance condition is satisfied.

We now prove the marginal consistency condition for pixel normalization. We can define the inverse of the operator $\mathcal{O}$ induced by PixelNorm, and since pixel normalization is performed on pixel level, it is easy to show that $\mathcal{O}^{-1}(\boldsymbol{x}(I) = \boldsymbol{b}) = \times_{i,j \in I} \mathcal{O}^{-1}(\boldsymbol{x}(i,j) = \boldsymbol{b}(i,j))$, with $\mathcal{O}^{-1}(\boldsymbol{x}(i,j) = \boldsymbol{b}(i,j)) = \{\boldsymbol{z}(i,j) = \alpha\boldsymbol{x}(i,j) + \beta, \forall \alpha, \beta \in \mathbb{R}\}$. This implies for any finite subsets $I' \subset I$ and the corresponding $\boldsymbol{b}' = \boldsymbol{b}(I')$, we have

$$
\begin{aligned}
P_{X,I}(\boldsymbol{x}(I') = \boldsymbol{b}') &= \int \int_{\mathcal{O}^{-1}(\boldsymbol{x}(I')=\boldsymbol{b}') \times \mathcal{O}^{-1}(\boldsymbol{x}(I \backslash I')=\boldsymbol{a})} P_{Z,I}(\boldsymbol{z}(I'), \boldsymbol{z}(I \backslash I')) \, d\boldsymbol{z}(I) d\boldsymbol{a} \\
&= \int \int_{\mathcal{O}^{-1}(\boldsymbol{x}(I')=\boldsymbol{b}') \times \mathcal{O}^{-1}(\boldsymbol{x}(I \backslash I')=\boldsymbol{a})} P_{Z,I}(\boldsymbol{z}(I'), \boldsymbol{z}(I \backslash I')) \, d\boldsymbol{a} d\boldsymbol{z}(I) \\
&= \int_{\mathcal{O}^{-1}(\boldsymbol{x}(I')=\boldsymbol{b}')} P_{Z,I}(\boldsymbol{z}(I'), \boldsymbol{z}(I \backslash I')) \, d\boldsymbol{z}(I) \\
&= \int_{\mathcal{O}^{-1}(\boldsymbol{x}(I')=\boldsymbol{b}')} P_{Z,I}(\boldsymbol{z}(I')) \, d\boldsymbol{z}(I') \\
&= P_{X,I'}(\boldsymbol{x}(I') = \boldsymbol{b}'),
\end{aligned}
$$

where in the second equality the exchange of integration follows Fubini's theorem, and the fourth equality follows from the assumption that $\boldsymbol{z}(\cdot, \cdot)$ is a stochastic process. Therefore we have proved the marginal consistency condition for $\boldsymbol{x}(\cdot, \cdot)$, and $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process. □

Lemma B.5. *For a bi-linear up-sampling operator with scale $U = 2$ and boundary cropping, $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$ if $\boldsymbol{z}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$.*

Proof. The bi-linear upsampling operator with scale U=2 and boundary cropping is given as follows:

$$\boldsymbol{x}(2h, 2w) = \frac{3}{4}\left(\frac{3}{4}\boldsymbol{z}(h,w) + \frac{1}{4}\boldsymbol{z}(h, w+1)\right) + \frac{1}{4}\left(\frac{3}{4}\boldsymbol{z}(h+1, w) + \frac{1}{4}\boldsymbol{z}(h+1, w+1)\right),$$

$$\boldsymbol{x}(2h, 2w+1) = \frac{3}{4}\left(\frac{1}{4}\boldsymbol{z}(h,w) + \frac{3}{4}\boldsymbol{z}(h, w+1)\right) + \frac{1}{4}\left(\frac{1}{4}\boldsymbol{z}(h+1, w) + \frac{3}{4}\boldsymbol{z}(h+1, w+1)\right),$$

$$\boldsymbol{x}(2h+1, 2w) = \frac{1}{4}\left(\frac{3}{4}\boldsymbol{z}(h,w) + \frac{1}{4}\boldsymbol{z}(h, w+1)\right) + \frac{3}{4}\left(\frac{3}{4}\boldsymbol{z}(h+1, w) + \frac{1}{4}\boldsymbol{z}(h+1, w+1)\right),$$

$$\boldsymbol{x}(2h+1, 2w+1) = \frac{1}{4}\left(\frac{1}{4}\boldsymbol{z}(h,w) + \frac{3}{4}\boldsymbol{z}(h, w+1)\right) + \frac{3}{4}\left(\frac{1}{4}\boldsymbol{z}(h+1, w) + \frac{3}{4}\boldsymbol{z}(h+1, w+1)\right).$$

Using Lemma B.1 we can show that $\{\boldsymbol{x}(2h, 2w) \mid (h, w) \in \mathbb{Z}^2\}$ is a stochastic process, and similarly the other three are stochastic processes as well. Also for any finite numbers H, W, if we write

$$S(H, W, 0, 0) = \sum_{h=-H}^{W}\sum_{w=-W}^{W} \boldsymbol{x}(2h, 2w)^2, \quad S(H, W, 0, 1) = \sum_{h=-H}^{W}\sum_{w=-W}^{W} \boldsymbol{x}(2h, 2w+1)^2,$$

$$S(H, W, 1, 0) = \sum_{h=-H}^{W}\sum_{w=-W}^{W} \boldsymbol{x}(2h+1, 2w)^2, \quad S(H, W, 1, 1) = \sum_{h=-H}^{W}\sum_{w=-W}^{W} \boldsymbol{x}(2h+1, 2w+1)^2,$$

then we have

$$\sqrt{\sum_{i=-2H}^{2H+1}\sum_{j=-2W}^{2W+1} \boldsymbol{x}(i,j)^2} \leq \sum_{i=0}^{1}\sum_{j=0}^{1} S(H, W, i, j) \leq 4\max_{i,j} S(H, W, i, j).$$

Also $S(H, W, i, j)$ is non-decreasing as H, W increase, so $S(H, W, 0, 0) \uparrow \|\boldsymbol{x}(2h, 2w)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2$ (similarly for the others). Therefore we have all $S(H, W, i, j), \forall i, j$ bounded by

$$\max_{i,j} S(H, W, i, j) \leq \max(\|\boldsymbol{x}(2h, 2w)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2, \|\boldsymbol{x}(2h, 2w+1)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2,$$

$$\|\boldsymbol{x}(2h+1, 2w)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2, \|\boldsymbol{x}(2h+1, 2w+1)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2) < +\infty,$$

where the last inequality comes from the fact that $\{\boldsymbol{x}(2h, 2w) \mid (h, w) \in \mathbb{Z}^2\}$ and the other three collections are stochastic processes on $\mathcal{L}^2(\mathbb{Z}^2)$. Therefore the supremum in below is also finite:

$$\sup_{H,W\to+\infty} \sqrt{\sum_{i=-2H}^{2H+1}\sum_{j=-2W}^{2W+1} \boldsymbol{x}(i,j)^2} \leq 4\max(\|\boldsymbol{x}(2h, 2w)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2, \|\boldsymbol{x}(2h, 2w+1)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2,$$

$$\|\boldsymbol{x}(2h+1, 2w)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2, \|\boldsymbol{x}(2h+1, 2w+1)\|_{\mathcal{L}^2(\mathbb{Z}^2)}^2).$$

Also we know the sum inside the square root on the LHS is non-decreasing when H, W increase. Therefore by the monotonic convergence theorem, the limit of the sum exists and

$$\|\boldsymbol{x}(\cdot, \cdot)\|_{\mathcal{L}^2(\mathbb{Z}^2)} = \lim_{H,W\to+\infty} \sqrt{\sum_{i=-2H}^{2H+1}\sum_{j=-2W}^{2W+1} \boldsymbol{x}(i,j)^2} < +\infty.$$

Therefore $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$. □

Lemma B.6. *For a nearest up-sampling operator with finite scale U, $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$ if $\boldsymbol{z}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$.*

Proof. The nearest up-sampling operator with finite scale U is given as

$$\boldsymbol{x}(Uh+i, Uw+j) = \boldsymbol{z}(h, w), \quad \forall(h, w) \in \mathbb{Z}^2, 0 \leq i < U, 0 \leq j < U.$$

Using similar ideas in the proof of Lemma B.5, one can show that $\boldsymbol{x}(\cdot, \cdot)$ is a stochastic process on $\mathcal{L}^2(\mathbb{Z}^2)$. □

B.2. Proofs for stationarity preserving

In below we show the stationarity preserving properties for the transformations used by an CNN. We say an operator $\mathcal{O}$ is *stationarity preserving*, if for any strictly stationary spatial process $z(\cdot, \cdot)$, the output process $\{x(i, j) = \mathcal{O}(z(\cdot, \cdot))(i, j)\}$ is strictly spatial stationary. Note that it suffices to investigate stationarity properties on patches derived from $x(\cdot, \cdot)$ as any finite subset of the index set $\mathbb{Z}^2$ is located within some finite-size patch. In the rest of the section we denote the distribution of any finite-size patch

Lemma B.7. *Convolution with finite filter size k, stride 1 and no padding preserves strict spatial stationarity.*

Proof. Again we consider w.l.o.g. a convolutional filter with filter size $K = 2K' + 1$ output channel 1. For any finite subset $I = \{1 : H\} \times \{1 : W\} \subset \mathbb{Z}^2$ we can "push" the boundary of I by size K' and obtain another index set J so that applying the convolution filter on $z(J)$ returns $x(I)$. Also define the inverse set of the operator on $x(I)$ as

$$\mathcal{O}^{-1}(x(I) = b) = \{a \in \mathbb{R}^{|J| \times C} \mid x(I) = \mathcal{O}(z(\cdot, \cdot))(I) = b, z(J) = a\}.$$

Similarly we can define such subset $\tilde{J}$ for the shifted set $\tilde{I} = \{(i + h, j + w) \mid (i, j) \in I\}$ given some shifting direction (h, w). Importantly, due to translation invarance of convolutions, $\tilde{J} = \{(i + h, j + w) \mid (i, j) \in J\}$ is also a shifted set of J with shifting direction (h, w). The shift invarance of convolution also means $\mathcal{O}^{-1}(x(I) = b) = \mathcal{O}^{-1}(x(\tilde{I}) = b)$. Furthermore the input process is stationary, i.e. $P_{Z,J}(z(J)) = P_{Z,\tilde{J}}(z(\tilde{J}))$. Putting these results together, we have

$$P_{X,I}(x(I) = b) = \int_{\mathcal{O}^{-1}(x(I)=b)} P_{Z,J}(z(J)) dz(J)$$

$$= \int_{\mathcal{O}^{-1}(x(\tilde{I})=b)} P_{Z,\tilde{J}}(z(\tilde{J})) dz(\tilde{J}) = P_{X,\tilde{I}}(x(\tilde{I}) = b).$$

Therefore $x(\cdot, \cdot)$ is a strictly spatial stationary process. $\square$

Lemma B.8. *Pixel normalization, and pixel-wise non-linear operator defined by an activation function $f(\cdot)$, preserves strict spatial stationarity.*

Proof. We prove this lemma for pixel-wise non-linear operators first, the proof for Pixel normalization can be done accordingly. Since the non-linearity $f(\cdot)$ is applied pixel-wise, this means

$$\mathcal{O}^{-1}(x(I) = b) = \times_{(i,j) \in I} f^{-1}(x(i, j) = b(i, j)) = \times_{(i,j) \in I} \{a \in \mathbb{R}^C \mid f(a) = b(i, j)\}.$$

Also it is straight-forward that $\mathcal{O}^{-1}(x(I) = b) = \mathcal{O}^{-1}(x(\tilde{I}) = b)$, and by assumption we have $P_{Z,J}(z(J)) = P_{Z,\tilde{J}}(z(\tilde{J}))$. Therefore one can show in similar way as to prove Lemma B.7 (with $J = I, \tilde{J} = \tilde{I}$) that $x(\cdot, \cdot)$ is a strictly spatial stationary process. $\square$

Lemma B.9. *Bi-linear upsampling with scale $U = 2$ and boundary cropping transforms a strictly stationary spatial process to a cyclostationary spatial process of period $(2, 2)$.*

Proof. It suffice to prove that $P_{X,I}(x(I) = b) = P_{X,\tilde{I}}(x(\tilde{I}) = b)$ for all

$$I = \{a, a + h\} \times \{b : b + w\}, \tilde{I} = \{a + 2, a + 2 + h\} \times \{b + 2 : b + 2 + w\}, \forall a, b \in \mathbb{Z}, h, w \in \mathbb{N}^+.$$

Since for any finite a, h there exist finite $H_{min} < H_{max}$ such that $\{a, a + h\} \subset \{2H_{min} : 2H_{max}\}$, we only need to show the equality for

$$I = \{2H_{min} : 2H_{max}\} \times \{2W_{min} : 2W_{max}\}, \forall H_{min} < H_{max}, W_{min} < W_{max} \in \mathbb{Z},$$

and the corresponding shifted index set is $\tilde{I} = \{2H_{min} + 2 : 2H_{max} + 2\} \times \{2W_{min} + 2 : 2W_{max} + 2\}$. From the operator presented in the proof of Lemma B.5 we see that $J = \{H_{min} : H_{max} + 1\} \times \{W_{min} : W_{max} + 1\}$, $\tilde{J} = \{H_{min} + 1 : H_{max} + 2\} \times \{W_{min} + 1, ..., W_{max} + 2\}$, and $\mathcal{O}^{-1}(x(I) = b) = \mathcal{O}^{-1}(x(\tilde{I}) = b)$. Therefore the equality holds given that $z(\cdot, \cdot)$ a strictly stationary process (using proof techniques presented in the proof of Lemma B.7). $\square$

Lemma B.10. *Nearest upsampling with scale $U = 2$ transforms a strictly stationary spatial process to a cyclostationary spatial process of period $(2, 2)$.*

Proof. Similar to the proof of Lemma B.9, it suffice to show $P_{X,I}(\boldsymbol{x}(I) = \boldsymbol{b}) = P_{X,\tilde{I}}(\boldsymbol{x}(\tilde{I}) = \boldsymbol{b})$ for $I = \{2H_{min} : 2H_{max}\} \times \{2W_{min} : 2W_{max}\}, \forall H_{min} < H_{max}, W_{min} < W_{max} \in \mathbb{Z}$, and the corresponding shifted index set is $\tilde{I} = \{2H_{min} + 2 : 2H_{max} + 2\} \times \{2W_{min} + 2 : 2W_{max} + 2\}$. From the operator presented in the proof of Lemma B.6 we see that $J = \{H_{min} : H_{max}\} \times \{W_{min} : W_{max}\}$, $\tilde{J} = \{H_{min} + 1 : H_{max} + 1\} \times \{W_{min} + 1, ..., W_{max} + 1\}$, and $\mathcal{O}^{-1}(\boldsymbol{x}(I) = \boldsymbol{b}) = \mathcal{O}^{-1}(\boldsymbol{x}(\tilde{I}) = \boldsymbol{b})$. Therefore the equality holds given that $\boldsymbol{z}(\cdot, \cdot)$ is a strictly stationary process (using proof techniques presented in the proof of Lemma B.7). $\square$

We note that similar proof techniques of Lemmas B.9 and B.10 can be applied to show the general result: these two consistent up-sampling methods increase the stationarity period by 2.

C. Related work: extended discussions

Generating high fidelity images State-of-the-art *un-conditional* GANs are able to generate high fidelity images of size 1024×1024 (Karras et al., 2018a;b; Brock et al., 2019). However these GAN models require full-scale images for training, and they can only generate images of the same *fixed* size as the training images. By contrast, ∞-GAN only takes small patches of different resolutions as training data, and the generator, once trained, can generate images of arbitrary size either in one shot or incrementally.

Other non-adversarial generative models have also been shown to be capable of high fidelity image generation (Kingma & Dhariwal, 2018; Huang et al., 2018; Menick & Kalchbrenner, 2019). Specifically, the Subscale Pixel Network (SPN) (Menick & Kalchbrenner, 2019), as the state-of-the-art auto-regressive generative model, uses a multi-scale ordering by sub-sampling pixel locations into rows and columns and progressing in raster ordering (Reed et al., 2017). This particular ordering requires a canvas of pre-defined size, making SPNs incapable of generating images of arbitrary/infinite sizes.

(Stationary) stochastic processes Auto-regressive processes, moving average processes and their combinations have been widely applied to spatial data modelling (see e.g. Huang, 1984; Haining, 1978; Ripley; Anselin, 2013). The ∞-GAN's generator can be viewed as an extension to the non-linear moving average model (Robinson, 1977); it is a consistent transform of stochastic processes, and the dependence between an image patch and the corresponding latent variables is independent with the location of the patch in that image.

Super resolution GANs have also been applied to super resolution, i.e. estimating a high resolution image from its low-resolution counterpart (Ledig et al., 2017; Sønderby et al., 2016; Wang et al., 2018). Our up-scaling network is similar to StackGAN++ (Zhang et al., 2018): both methods train a stack of generators (with the pre-image outputs from the last generator as the inputs) on multi-resolution images (Denton et al., 2015; Zhang et al., 2017). Still all existing methods including StackGAN++ require full images for training, while our up-scaling network is trained on patches.

Texture synthesis Give a reference texture image, model-free texture synthesis techniques perform *conditional* generation by re-sampling pixels or patches from the original texture (see e.g. Barnes et al., 2009; Efros & Leung, 1999; Wei & Levoy, 2000; Efros & Freeman, 2001; Kwatra et al., 2003). Perhaps PatchMatch (Barnes et al., 2009) is the closest related approach, which fills in the missing values by a fast nearest neighbour search over patches from a reference image.

Model-based approaches that use CNNs have recently become popular for texture synthesis. With a pre-trained network as texture feature extractor (Gatys et al., 2015b;a; Johnson et al., 2016), recent approaches used (conditional) GANs to generate texture images with features that are similar to the features from either a reference input (Ulyanov et al., 2016a; Li & Wand, 2016), or from a pre-defined set known as style bank (Ulyanov et al., 2017). The closely related approach is the (Periodic) Spatial GAN (PSGAN) (Bergmann et al., 2017; Jetchev et al., 2016), however it uses an inconsistent generator, and it requires large images for learning the stationarity pattern. By contrast, our model only sees small patches, and the dependency between patches is induced by the overlapping pixels in $\boldsymbol{z}$ space.

Terrain/map generation Traditionally terrain/map generation for video games are dominated by procedural generation techniques, e.g. procedural noise functions (Perlin, 1985; 2002; Fournier et al., 1982) and physical process simulations (Musgrave et al., 1989; Olsen, 2004). A recent attempt to apply GANs to world map generation is presented in (Beckham &
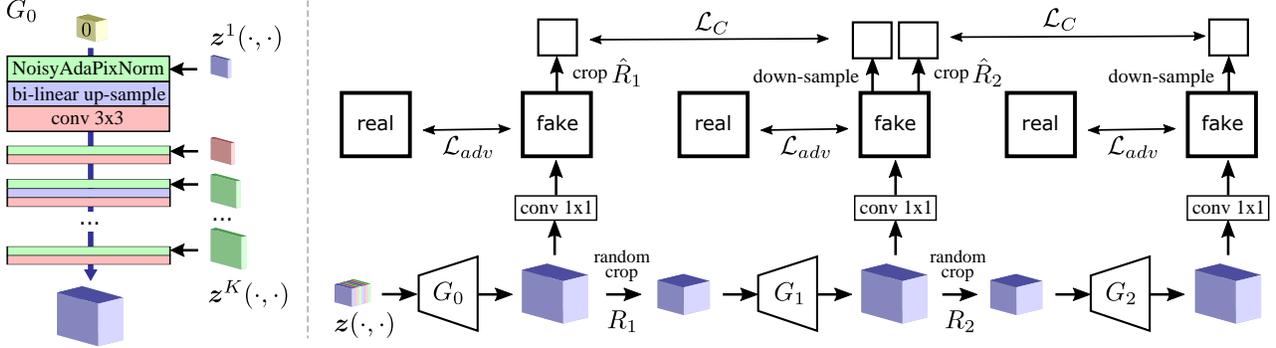
Figure 4. The network architecture ($L = 2$) and the training mechanism.

Pal, 2017), which first trained a DCGAN (Radford et al., 2015) on 512×512 crops from the NASA height map data of the Earth, then applied pix2pix (Isola et al., 2017) to paint the terrain texture conditioned on the height map. This approach uses vector inputs and inconsistent CNNs, thus it can only generate terrains of size 512×512.

D. Infinite GAN

As motivated by the theoretical results above, we design ∞-GAN, whose generator is well-defined in terms of our theory, to allow generation of images at different *resolutions* and in arbitrary image *sizes*.[1] To enable efficient training on vary large images, we follow the multi-scale training strategy (Denton et al., 2015; Zhang et al., 2017; 2018; Karras et al., 2018a) to construct a stack of $L + 1$ generators $\{G_0(\cdot), ..., G_L(\cdot)\}$, each of them transforms a spatial stochastic process to another spatial stochastic process. Practically, this means given a sample of the latent tensor $\boldsymbol{z}(J)$, ∞-GAN is capable of generating a set of images $\{\boldsymbol{x}_0(I_0), ..., \boldsymbol{x}_L(I_L)\}$ of different resolutions. Here I_l is the corresponding pixel index set of the lth generator's image output. We name the first generator $G_0(\cdot)$ as the *extension network* of ∞-GAN's generators, which is mainly responsible for learning the dependency structure in data that is crucial for incrementally extending images to larger *sizes*. As the other networks $G_l(\cdot), l \geq 1$ in the hierarchy are used for up-scaling the *resolution* of the images, these networks are referred to as the *up-scaling networks*.

The extension network The extension network $G_0(\cdot)$ takes in as inputs finite size samples from multiple latent processes $\{\boldsymbol{z}^1(\cdot, \cdot), ..., \boldsymbol{z}^K(\cdot, \cdot)\}$, and produces a *pre-image feature tensor* $\boldsymbol{f}_0(I_0)$ that is later transformed into an image patch $\boldsymbol{x}_0(I_0)$ with 1×1 convolution (Zhang et al., 2018; Karras et al., 2018a). See Figure 4 for a visualization, in detail, we construct $G_0(\cdot)$ with K blocks of {Noisy AdaPixNorm, Conv 3×3} layers (possibly with up-sampling layers in the blocks), and the convolutional layers use stride 1 and no padding. The Noisy AdaPixNorm replaces instance normalization (Ulyanov et al., 2016b) in StyleGAN's noise-in AdaIN layers (Karras et al., 2018b) with pixel normalization.[2] Therefore with the inputs $\{\mathbf{h}(i, j) \in \mathbb{R}^C\}$ and the latent tensor values $\{\boldsymbol{z}^k(i, j) \in \mathbb{R}\}$, the kth Noisy AdaPixNorm layer's output is $\mathbf{y}(i, j) = \boldsymbol{\beta}_k \odot \text{PixelNorm}(\mathbf{h}(i, j) + \boldsymbol{z}^k(i, j)\boldsymbol{w}_k) + \boldsymbol{\gamma}_k$, where $\boldsymbol{\beta}_k, \boldsymbol{\gamma}_k, \boldsymbol{w}_k \in \mathbb{R}^C$. For training, we apply the Wasserstein GAN approach with gradient penalty (WGAN-GP) (Gulrajani et al., 2017), using real image *patches* sampled from the data $\mathcal{D}_0$ and the generator:

$$\mathcal{L}_{adv}(G_0, D_0) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_0}[D_0(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim P_{X_0}}[D_0(\mathbf{x})] + \lambda \mathbb{E}_{\mathbf{x} \sim \tilde{P}_{X_0}}[(\|\nabla_{\mathbf{x}} D_0(\mathbf{x})\| - 1)^2]. \tag{3}$$

Here the image patches in $\mathcal{D}_0$ are generated by random cropping from a very large training image. The patch size of crops needs to be selected carefully. Assuming the extension network contains K up-sampling layers with scale 2, this means the *model patch*, defined as the projected field in $\boldsymbol{x}_0$ space for a pixel in $\boldsymbol{z}^1$ space, has size $(2^K, 2^K)$. Also due to the usage of consistent operations, it requires the latent tensor in $\boldsymbol{z}^1$ space to have spatial size at least (h, w) in order to generate a model patch in $\boldsymbol{x}_0$ space. Assuming i.i.d. Gaussian variables for all latent processes, this means the output process $\boldsymbol{x}_0(\cdot, \cdot)$ has stationarity period $(2^K, 2^K)$, and it is $(2^K h, 2^K w)$-dependent in the sense that any two pixels $\boldsymbol{x}_0(i_1, j_1)$ and $\boldsymbol{x}_0(i_2, j_2)$

---

[1]Note the difference between image resolution and image size.

[2]Even when the input stochastic process is stationary and ergodic, instance norm's estimation of ensemble mean and variance across pixel indices can vary a lot depending on the image size.

with $|i_1 - i_2| > 2^K h$ and/or $|j_1 - j_2| > 2^K w$ are independent. This means one can select the patch size (H, W) of the training images as $(2^{K+1}, 2^{K+1}) \leq (H, W) \leq (2^K h, 2^K w)$ in order to learn both the marginal distribution over a model patch, as well as the dependencies between model patches. Empirically we select $(H, W) = (2^{K+1}, 2^{K+1})$, which means in training the spatial size of the latent tensor in z^1 space is $(h + 1, w + 1)$.

The up-scaling networks Each up-scaling network $G_l(\cdot)$ takes the pre-image output $f_{l-1}(J_l)$ from the last generator with some index set J_l, and produces a feature tensor $f_l(I_l)$ with larger spatial size by consistent up-sampling and convolution. This feature tensor is then transformed into the lth layer's image output $x_l(I_l)$ by a 1×1 convolution.

For training, apart from the WGAN-GP loss (3), up-scaling consistency loss is applied to enforce the alignment of images in the multi-scale hierarchy. However, with up-sampling layers in use, the image size of $x(I_l)$ grows exponentially. We introduce an efficient training method for the up-scaling networks $G_l(\cdot), l = 1, ..., L$, by using image patches of fixed size at all resolutions in the hierarchy. Assume that all the generated images during training have fixed size (H, W). Then one can perform a *random crop* operation with index set $J_l = R_l(I_{l-1})$ on the feature tensor $f_{l-1}(I_{l-1})$ to obtain the input $f_{l-1}(J_l)$ for $G_l(\cdot)$, and the cropping operator R_l is selected so that the generated image $x_l(I_l)$ has the desired size (H, W). Later $x_l(I_l)$ is down-sampled (with method $d(\cdot)$) to match the resolution of x_{l-1}, and we crop the corresponding low-resolution patch from $x_{l-1}(I_{l-1})$ with index set $\hat{I}_{l-1} = \hat{R}_l(I_{l-1})$ to compute the consistency loss between images of different resolutions:

$$\mathcal{L}_C(G_l) = \mathbb{E}_{x_{l-1} \sim P_{X_{l-1}}} \mathbb{E}_{x_l \sim P_{X_l}} \left[||x_{l-1}(\hat{I}_{l-1}) - d(x_l(I_l))||_2^2 \right]. \tag{4}$$

Since the up-scaling network uses consistent up-sampling methods, the zoom-in ratio $|I_l|/|J_l|$ is not necessarily a power of two. Therefore we define $\hat{R}_l$ such that $x_{l-1}(\hat{I}_{l-1})$ is the centered $(H/2, W/2)$ patch within $x_{l-1}(J_l)$. In sum, the total loss for training the up-scaling network is:

$$\mathcal{L}(G_l, D_l) = \mathcal{L}_{adv}(G_l, D_l) + \lambda_l \mathcal{L}_C(G_l). \tag{5}$$

See Figure 4 for a visualization. We further add a consistency loss between images generated by G_0 and down-sampled images from G_L, which we empirically find to improve results. The images patches in $\mathcal{D}_l$ are again generated by random crops from real images, and for all $l \geq 0$ the image patches have the same fixed size. It also means the discriminator D_l used in $\mathcal{L}_{adv}$ sees small patches only, unlike previous multi-scale training methods (Zhang et al., 2018; Karras et al., 2018a; Denton et al., 2015) where the discriminators in the hierarchy observe images of exponentially increasing size.

E. Additional experiments

E.1. Qualitative evaluations on texture and panoramic data

Texture generation We train the ∞-GAN model on 4 texture images. The full-scale data has size 512×512 (see the first column in Figure 5), and the three datasets $\mathcal{D}_0, \mathcal{D}_1$ and $\mathcal{D}_2$ are constructed in a similar way as done in the world generation task. The third column of Figure 5 shows that the ∞-GAN has captured the texture features, even when the network has never observed 256×256 image patches with the full-scale resolution. We further qualitatively evaluate the 2048×2048 images generated by ∞-GAN (the last 4 columns); for reference we tiled the original images to the same size, which is shown in the second column of Figure 5. It is clear that the ∞-GAN model, after training, is capable of generating both realistic looking and non-repeating texture images.

Panoramic city view Lastly we train the extension network $G_0(\cdot)$ of the ∞-GAN on a panoramic image of the New York city, and then use the model to extend the landscape view horizontally. The training data and the generated samples are shown in Figure 6. We see that the ∞-GAN model is able to generate city views of different patterns (e.g. with many skyscrapers and/or harbour views).

F. Experimental details

F.1. Data collection

Texture We download from https://www.textures.com/ 4 texture images. Then we construct datasets from each of them. Specifically, the original image is referred to as "texture 4x", and we down-sized this image to "texture 2x" and "texture 1x". Then random cropping is applied to each of the 4 images to obtain datasets $\{\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2\}$ of increasing resolutions, each of them contains $(64, 64)$ image patches of the corresponding resolution.
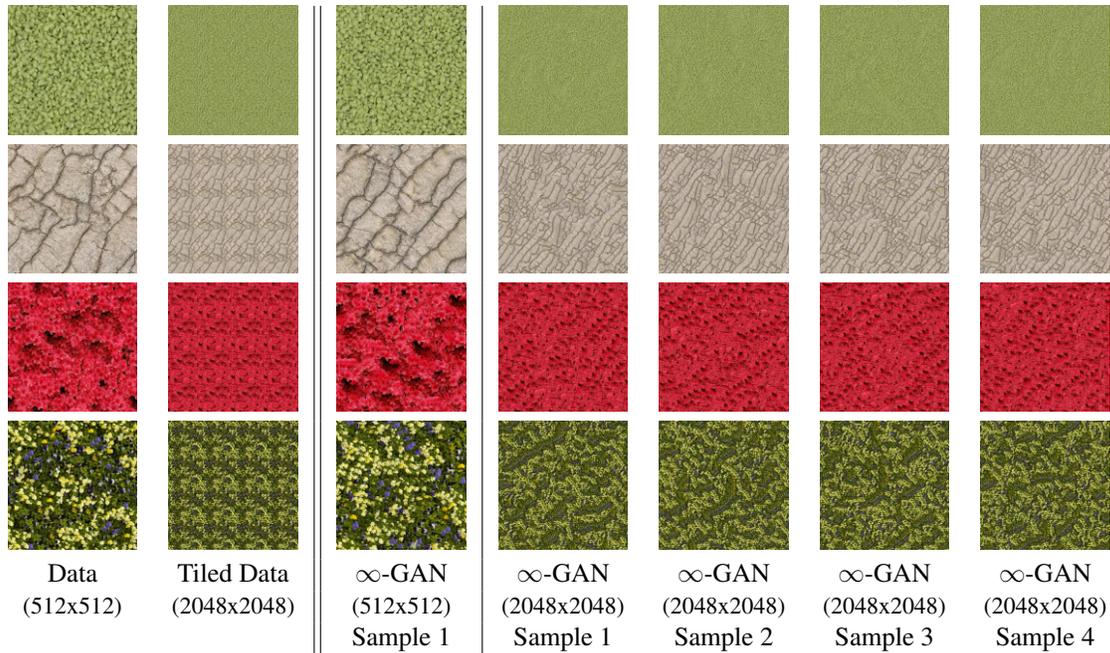
Data (512x512)	Tiled Data (2048x2048)	∞-GAN (512x512) Sample 1	∞-GAN (2048x2048) Sample 1	∞-GAN (2048x2048) Sample 2	∞-GAN (2048x2048) Sample 3	∞-GAN (2048x2048) Sample 4

Figure 5. Visualizing generated textures images. The first two columns show the training data and its tiled version. The third column shows generated images at the same scale as the input, and the last four columns show the huge samples as compared to the tiled training images. Best view in color.
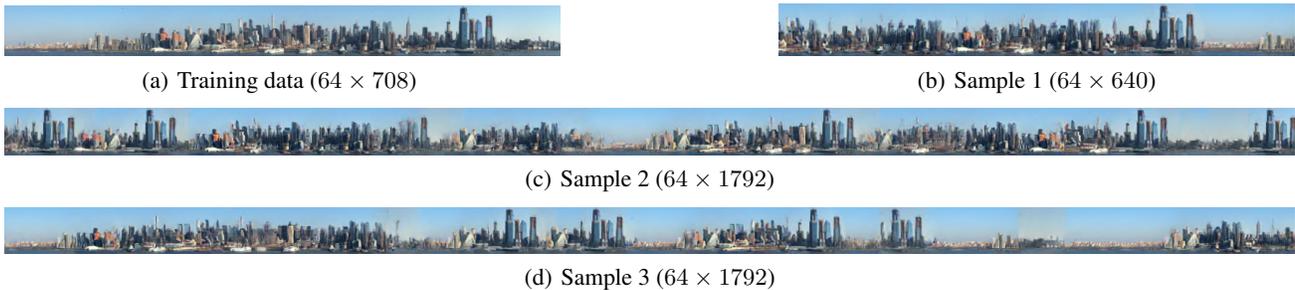

(a) Training data (64×708)

(b) Sample 1 (64×640)

(c) Sample 2 (64×1792)

(d) Sample 3 (64×1792)

Figure 6. Generated panoramic views. The model only sees 64×64 data patches during training.

Panoramic landscape view   We download the panoramic landscape images from

https://en.wikipedia.org/wiki/Skyline#/media/File:10_mile_panorama_of_NYC,_Feb.,_2018.jpg

We removed the numbers and then cropped the image to remove part of the sky. We further down-sized the image to have vertical spatial size 64. Then the dataset $\mathcal{D}_0$ contains $(64, 64)$ image patches which are randomly cropped from the full-scale image.

Satellite world map   The full image is downloaded from:

https://eoimages.gsfc.nasa.gov/images/imagerecords/74000/74117/world.200408.3x5400x2700.png

We remove the bottom pixels containing Antarctica in order to remove the bias introduced by the Mercator projection and then down-sized it to half of the size in each spatial dimension. Then multi-scale down-sizing and random cropping are applied to each of the 4 images to obtain datasets $\{\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2\}$ of increasing resolutions, each of them contains $(64, 64)$ image patches of the corresponding resolution.

F.2. Network architectures and training details

Network architecture for $G_0(\cdot)$ For example in training time we start from an $6 \times 6 \times 128$ input of zero values. Note here that all the convolutional layers use stride 1 and no padding. The latent tensors $\{z^k(i,j)\}$ are i.i.d. Gaussian noises during training, which is sampled when computing a Noisy AdaPixNorm (NAPN) operation. They have the same spatial shape as the input to NAPN but with channel 1, e.g. if the input tensor has shape $(6, 6, C)$, then the sampled z^k tensor will have shape $(6, 6, 1)$. See main text for the math expression of the NAPN operation. We use bi-linear interpolation for up-sampling, to remain consistent we crop out the boundary pixels with edge size 1. This means that a tensor of shape (H, W, C), after this up-sampling, will be transformed to another tensor of shape $(2H - 2, 2W - 2, C)$.

With input tensor of shape $(H, W, C) = (6, 6, 128)$:

Block 1:

$$(6, 6, 128) \text{ input} \rightarrow \text{NAPN} \rightarrow \text{bi-linear up-sample} \rightarrow \text{conv } 3 \times 3 \rightarrow \text{ReLU} \rightarrow \text{output } (8, 8, 128).$$

Block 2:

$$(8, 8, 128) \text{ input} \rightarrow \text{NAPN} \rightarrow \text{bi-linear up-sample} \rightarrow \text{conv } 3 \times 3 \rightarrow \text{ReLU} \rightarrow \text{output } (12, 12, 128).$$

Block 3:

$$(12, 12, 128) \text{ input} \rightarrow \text{NAPN} \rightarrow \text{bi-linear up-sample} \rightarrow \text{conv } 3 \times 3 \rightarrow \text{ReLU} \rightarrow \text{output } (20, 20, 128).$$

Block 4:

$$(20, 20, 128) \text{ input} \rightarrow \text{NAPN} \rightarrow \text{bi-linear up-sample} \rightarrow \text{conv } 3 \times 3 \rightarrow \text{ReLU} \rightarrow \text{output } (36, 36, 128).$$

Block 5:

$$(36, 36, 128) \text{ input} \rightarrow \text{NAPN} \rightarrow \text{conv } 3 \times 3 \rightarrow \text{ReLU} \rightarrow \text{output } (34, 34, 64).$$

Block 6:

$$(34, 34, 64) \text{ input} \rightarrow \text{NAPN} \rightarrow \text{bi-linear up-sample} \rightarrow \text{conv } 3 \times 3 \rightarrow \text{ReLU} \rightarrow \text{output } (64, 64, 64).$$

These blocks transforms the $(6, 6, 128)$ input tensor to $\boldsymbol{f}_0(I_0)$ of shape $(64, 64, 64)$. Lastly a 1×1 convolution and a Tanh layer are applied to $\boldsymbol{f}_0(I_0)$ to transform it into an image $\boldsymbol{x}_0(I_0)$ of shape $(64, 64, 3)$.

Note that batch-normalization (BN) (Ioffe & Szegedy, 2015) layers might be added before ReLU but after the conv 3×3 layers. Since in test time the "evaluation mode" of BN is used, then the applied normalization statistics are independent to the current latent tensors, therefore this test-time BN is still a consistent transformation of stochastic processes.

Network architecture for $G_l(\cdot), l \geq 1$ For example with an input $\boldsymbol{f}_{l-l}(J_l)$ of shape $(34, 34, 64)$ which is cropped from the output of the last network $\boldsymbol{f}_{l-l}(I_{l-1})$:

$$(34, 34, 64) \text{ input} \rightarrow \text{bi-linear up-sample} \rightarrow \text{conv } 3 \times 3 \rightarrow \text{ReLU} \rightarrow \text{output } (64, 64, 64).$$

These layers transforms the $(34, 34, 64)$ input tensor $\boldsymbol{f}_{l-l}(J_l)$ to $\boldsymbol{f}_l(I_l)$ of shape $(64, 64, 64)$. Lastly a 1×1 convolution and a Tanh layer are applied to $\boldsymbol{f}_l(I_l)$ to transform it into an image $\boldsymbol{x}_l(I_l)$ of shape $(64, 64, 3)$.

Network architecture for $D_l(\cdot), l \geq 0$ The discriminators at all layers use the same architecture, and they observes image inputs of the same size (in our case $(64, 64, 3)$). With an input $\mathbf{x}$ of shape $(64, 64, 3)$, a discriminator computes the scalar output by the following transforms. The architecture for each of the discriminators has the following architecture.

Layer 1:

$$(64, 64, 3) \text{ input} \rightarrow \text{ conv } 3 \times 3, \text{ stride 2, zero padding size 1} \rightarrow \text{LeakyReLU } (0.2) \rightarrow (32, 32, 64)$$

Layer 2:

$$(32, 32, 64) \text{ input} \rightarrow \text{ conv } 3 \times 3, \text{ stride 2, zero padding size 1} \rightarrow \text{LeakyReLU } (0.2) \rightarrow (16, 16, 64)$$

Layer 3:

$$(16, 16, 64) \text{ input} \rightarrow \text{ conv } 3 \times 3, \text{ stride 2, zero padding size } 1 \rightarrow \text{LeakyReLU } (0.2) \rightarrow (8, 8, 128)$$

Layer 4:

$$(8, 8, 128) \text{ input} \rightarrow \text{ conv } 3 \times 3, \text{ stride 2, zero padding size } 1 \rightarrow \text{LeakyReLU } (0.2) \rightarrow (4, 4, 128)$$

Layer 5:

$$(4, 4, 128) \text{ input} \rightarrow \text{ conv } 4 \times 4, \text{ stride 1, no padding} \rightarrow \text{LeakyReLU } (0.2) \rightarrow (1, 1, 128)$$

Layer 6:

$$(1, 1, 128) \text{ input} \rightarrow \text{ conv } 1 \times 1, \text{ stride 1, no padding} \rightarrow (1, 1, 1)$$

An important note   The index sets $\{J_l\}$ and $\{I_l\}$ are used only for mathematical rigor, none of the generative networks $G_i(\cdot)$ nor the discriminative networks $D_i(\cdot)$ take these index sets as inputs.

The hyper-parameters   We set the λ_l parameters associated with the consistency losses $\mathcal{L}_C(G_l)$ as $\lambda_1 = 1000, \lambda_2 = 1000$. As we also use the consistency loss to match between a patch cropped from $x_0(I_0)$ and a down-sampled version of $x_2(I_2)$, the balancing parameter for this consistency loss is selected as 16000. Average pooling is used as the down-sampling method $d(\cdot)$. The WGAN-GP losses $\mathcal{L}_{adv}(G_l, D_l)$ use $\lambda = 10$ for the gradient penalty. The training schedule is to optimize the parameters of $\{D_l(\cdot)\}$ for 5 iterations then optimize the parameters of $\{G_l(\cdot)\}$. We use Adam optimizer fot both the generators and the discriminators is selected, with learning rate 0.0001 and the momentum damping rate $\beta_1 = 0.5$.

Baseline   We use a PyTorch reimplementation of Spatial GAN and PSGAN which is linked to from the original authors repository.[3] We perform a grid search over hyperparameter settings for the baseline. We search find channel sizes for the generator and the discriminator in $[10, 20, 30, 40, 50, 60, 70, 80]$ and search input crop sizes in $[128, 256, 512, 1024]$. We choose the best architecture based on FID score. For spatial GAN this was a final channel size of 20 and a input crop size of 256. For PSGAN this was a final channel size of 40 and an input crop size of 512.

---

[3]https://github.com/zalandoresearch/famos