Aligning Latent and Image Spaces to Connect the Unconnectable

Ivan Skorokhodov[1], Grigorii Sotnikov[2,3,4], Mohamed Elhoseiny[1]
[1]King Abdullah University of Science and Technology (KAUST)
[2]Gradient, [3]Higher School of Economics, [4]Skolkovo Institute of Science and Technology
ivan.skorokhodov@kaust.edu.sa gdsotnikov@edu.hse.ru mohamed.elhoseiny@kaust.edu.sa

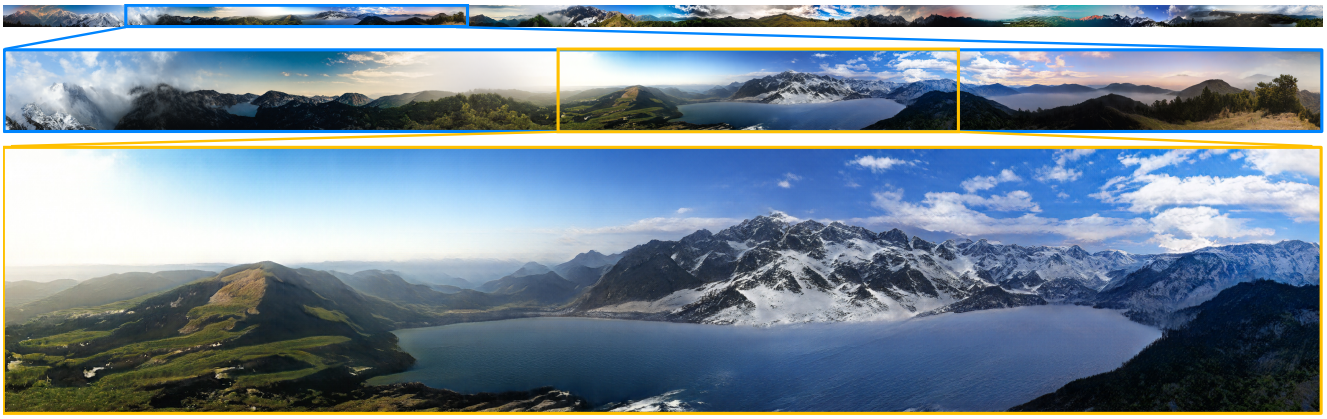


Figure 1: Our method can generate infinite images of diverse and complex scenes that transition naturally from one into another. It does so without any conditioning and trains without any supervision from a dataset of unrelated square images.

Abstract

In this work, we develop a method to generate infinite high-resolution images with diverse and complex content. It is based on a perfectly equivariant patch-wise generator with synchronous interpolations in the image and latent spaces. Latent codes, when sampled, are positioned on the coordinate grid, and each pixel is computed from an interpolation of the neighboring codes. We modify the AdaIN mechanism to work in such a setup and train a GAN model to generate images positioned between any two latent vectors. At test time, this allows for generating infinitely large images of diverse scenes that transition naturally from one into another. Apart from that, we introduce LHQ: a new dataset of 90k high-resolution nature landscapes. We test the approach on LHQ, LSUN Tower and LSUN Bridge and outperform the baselines by at least 4 times in terms of quality and diversity of the produced infinite images. The project website is located at https://universome.github.io/alis.

1. Introduction

Modern image generators are typically designed to synthesize pictures of some fixed size and aspect ratio. The real world, however, continues outside the boundaries of any captured photograph, and so to match this behavior, several recent works develop architectures to produce infinitely large images [29, 10, 57], or images that partially extrapolate outside their boundary [25, 43, 57].

Most of the prior work on infinite image generation focused on the synthesis of homogeneous texture-like patterns [20, 2, 29] and did not explore the infinite generation of complex scenes, like nature or city landscapes. The critical challenge of generating such images compared to texture synthesis is making the produced frames globally consistent with one another: when a scene spans across several frames, they should all be conditioned on some shared information. To our knowledge, the existing works explored three ways to achieve this: 1) fit a separate model per scene, so the shared information is encoded in the model's weights (e.g., [29, 40, 53, 11]); 2) condition the whole generation process on a global latent vector [57, 25, 43]; and 3) predict spatial latent codes autoregressively [10].

The first approach requires having a large-resolution photograph (like satellite images) and produces pictures whose variations in style and semantics are limited to the given imagery. The second solution can only perform some limited extrapolation since using a single global latent code

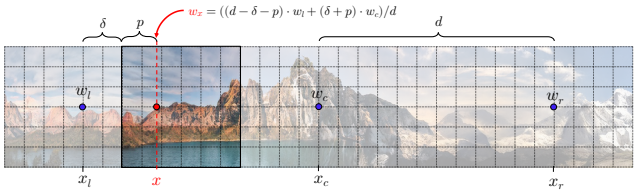


Figure 2: Illustration of our alignment procedure. We position initial latent codes (*anchors*) on the 2D coordinate space and compute latent code $\boldsymbol{w}_x$ for each position x as a linear interpolation between its two neighboring anchors.

cannot encompass the diversity of an infinite scenery (as also confirmed by our experiments). The third approach is the most recent and principled one, but the autoregressive inference is dramatically slow [15]: generating a single 256^2 image with [10]'s method takes us $\sim$10 seconds on a single V100 GPU.

This work, like [10], also seeks to build a model with global consistency and diversity in its generation process. However, in contrast to [10], we approach the problem from a different angle. Instead of slowly generating *local* latent codes autoregressively (to make them coherent with one another), we produce several *global* latent codes independently and train the generator to connect them.

We build on top of the recently proposed coordinate-based decoders that produce images based on pixels' coordinate locations [28, 25, 43, 1, 5] and develop the above idea in the following way. Latent codes, when sampled, are positioned on the 2D coordinate space — they constitute the "anchors" of the generation process. Next, each image patch is computed independently from the rest of the image, and the latent vector used for its generation is produced by linear interpolation between nearby anchors. If the distance $d \in \mathbb{R}$ between the anchors is sufficiently large, they cover large regions of space with the common global context and make the neighboring frames in these regions be semantically coherent. This idea of aligning latent and image spaces (ALIS) is illustrated in Figure 2. An important ingredient of our setup is the use of Fourier coordinate embeddings [42, 45], which provide crucial positional information and help to model high-frequency details. We inherit their design from INR-GAN [43].

Our model is GAN-based [14] and the generator is trained to produce plausible images from any position in space. This is in high contrast to existing coordinate-based approaches that generate samples only from $[0, 1]^2$ coordinates area, which constitutes a single frame size. To make the model generalize to any position in space, we do not input global coordinates information. Instead, we input only its position relative to the neighboring anchors, which, in turn, could be located arbitrarily on the x-axis.

We utilize StyleGAN2 architecture [24] for our method and modify only its generator component. Originally, StyleGAN2 passes latent vectors into the decoder by modulating and demodulating convolutional weights — an adjustment of adaptive instance normalization layer (AdaIN) [18]. For our generator, we redesign AdaIN to make it work with the coordinate-based latent vectors and develop Spatially-Aligned AdaIN (SA-AdaIN), described in Sec 3. Our model is trained in a completely unsupervised way from a dataset of unrelated square image crops, i.e. it never sees full panorama images or even different parts of the same panorama during training. By training it to produce realistic images located between arbitrary anchors describing different content (for example, mountains and a forest), it learns to connect unrelated scenes into a single panorama. This task can be seen as learning to generate camera transitions between two viewpoints located at semantically very different locations.

We test our approach on several LSUN categories and Landscapes HQ (LHQ): a new dataset consisting of 90k high-resolution nature landscape images that we introduce in this work. We outperform the existing baselines for all the datasets in terms of infinite image quality by at least 4 times and at least 30% in generation speed.

2. Related work

Coordinates conditioning. Coordinates conditioning is the most popular among the NeRF-based [33, 30] and occupancy-modeling [32, 6, 26] methods. [35, 4, 41] trained a coordinate-based generator that models a volume which is then rendered and passed to a discriminator. However, several recent works demonstrated that providing positional information can help the 2D world as well. For example, it can improve the performance on several benchmarks [28, 1] and lead to the emergence of some attractive properties like extrapolation [25, 43] or super-resolution [43, 5]. An important question in designing coordinate-based methods is how to embed positional information into a model [50, 13]. Most of the works rely either on raw coordinates [28, 57] or periodic embeddings with log-linearly distributed frequencies [33]. [42, 45] recently showed that using Gaussian distributed frequencies is a more principled approach. [38] developed a progressive growing technique for positional embeddings.

Infinite image generation. Existing works on infinite image generation mainly consider the generation of only texture-like and pattern-like images [20, 2, 11, 29, 39], making it similar to procedural generation [37, 34]. SinGAN [40] trains a GAN model on a single image and is able to produce its variations. Generating infinite images with diverse, complex and globally coherent content is a more intricate task since one needs to seamlessly connect both local and global features. LocoGAN [57] and Taming Transformers (TT) [10] generate images with arbitrary aspect ra-

tio and hence can be used for infinite image generation. LocoGAN [57] uses a single global latent code, which leads to content repetition (see Fig 7). TT [10] produces latent codes autoregressively which makes the generated content more diverse, but at the expense of being slower at test-time. [27] proposes both a model and a dataset to generate a sequence of images along a camera trajectory from a single RGB frame. Our approach shares some resemblance to image stitching [44], but instead of concatenating existing images without seams, we generate from scratch an *infinite* panorama. [21] constructed an infinite image by performing image retrieval + stitching from a vast image collection.

Image extrapolation. Another close line of research is image extrapolation (or image "outpainting" [54, 49]), i.e., predicting the surrounding context of an image given only its part. The latest approaches in this field rely on using GANs to predict an outlying image patch [16, 46, 51]. The fundamental difference of these methods compared to our problem design is the reliance on an input image as a starting point of the generation process.

Adaptive Normalization. Instance-based normalization techniques were initially developed in the style transfer literature [12]. Instance Normalization [47] was proposed to improve feed-forward style transfer [48] by replacing content image statistics with the style image ones. CIN [9] learned separate scaling and shifting parameters for each style. AdaIN [18] developed the idea further and used shift and scaling values produced by a separate module to perform style transfer from an arbitrary image. Similar to StyleGAN [23], we use AdaIN [18] without shifting to input latent information to the generator. But in contrast to its setup, we compute the scale weights by interpolating nearby latent codes using their coordinate positions instead of using global ones for the whole image.

Equivariant models. [55] added averaging operation to a convolutional block to improve its invariance/equivariance to shifts. [36] explored natural equivariance properties inside the existing classifiers. [8] developed a convolutional module equivariant to sphere rotations, and [7] generalized it to other symmetries. In our case, we manually construct a model to be equivariant to shifts in the coordinate space.

3. Method

We build upon StyleGAN2 [24] architecture and modify only its generator G as illustrated in Fig 4. All the other components, including discriminator D, the loss terms, and the optimization procedure, are left untouched.

To produce an image with our generator, we first need to sample *anchors*: latent codes that define the context in the space region in which the image is located. In this work, we consider only horizontal infinite image generation. Thus we need only three anchors to define the context: left anchor $\boldsymbol{w}_l$, center anchor $\boldsymbol{w}_c$ and right anchor $\boldsymbol{w}_r$. Following StyleGAN2, we produce a latent code $\boldsymbol{w}$ with the mapping network $\boldsymbol{w} \sim \mathsf{F}(\boldsymbol{z})$ where $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{d_z})$.

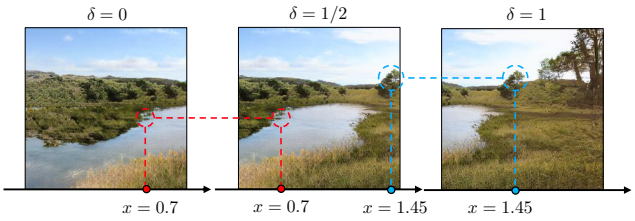


Figure 3: Our generator has the equivariance property by construction: we depict three samples with the coordinate shifts of 0, $1/2$ and 1, respectively, and this makes the resulted output move accordingly. As highlighted by dash circles, pixel values are equal when their coordinates are equal (up to numerical precision) for different samples. Samples remain of the same quality and diversity for any shift $s \in (-\infty, \infty)$.

To generate an image, we first need to define its location in space. It is defined relative to the left anchor $\boldsymbol{w}_l$ by a position of its left border $\delta \in [0, 2 \cdot d - W]$, where d is the distance between anchors and W is the frame width. In this way, δ gives flexibility to position the image anywhere between $\boldsymbol{w}_l$ and $\boldsymbol{w}_r$ anchors in such a way that it lies entirely inside the region controlled by the anchors. During training, $\boldsymbol{w}_l, \boldsymbol{w}_c, \boldsymbol{w}_r$ are positioned in the locations $-d, 0, d$ respectively along the x-axis and δ is sampled randomly. At test time, we position anchors $\boldsymbol{w}_i$ at positions $0, d, 2d, 3d, \ldots$ and move with the step size of $\delta = W$ along the x-axis while generating new images. This is illustrated in Figure 5.

Traditional StyleGAN2 inputs a latent code into the decoder via an AdaIN-like [18] weight demodulation mechanism, which is not suited for our setup since we use different latent codes depending on a feature coordinate position. This forces us to modify it into Spatially-Aligned AdaIN (SA-AdaIN), which we describe in Sec 3.2.

Our generator architecture is coordinate-based and, inspired by [25], we generate an image as 16 independent vertical patches, which are then concatenated together (see B for the illustration). Independent generation is needed to make the generator learn how to stitch nearby patches using the coordinates information: at test-time, it will be stitching together an infinite amount of them. Such a design also gives rise to an attractive property: spatial equivariance, that we illustrate in Figure 3. It arises from the fact that each patch does not depend on nearby patches, but only on the anchors $\boldsymbol{w}_l, \boldsymbol{w}_c, \boldsymbol{w}_r$ and its relative coordinates position δ. At each generator block, we concatenate coordinates information, represented as Fourier positional embeddings [42, 45]. We inherit coordinate layers without changes from the INR-GAN [43] repository.

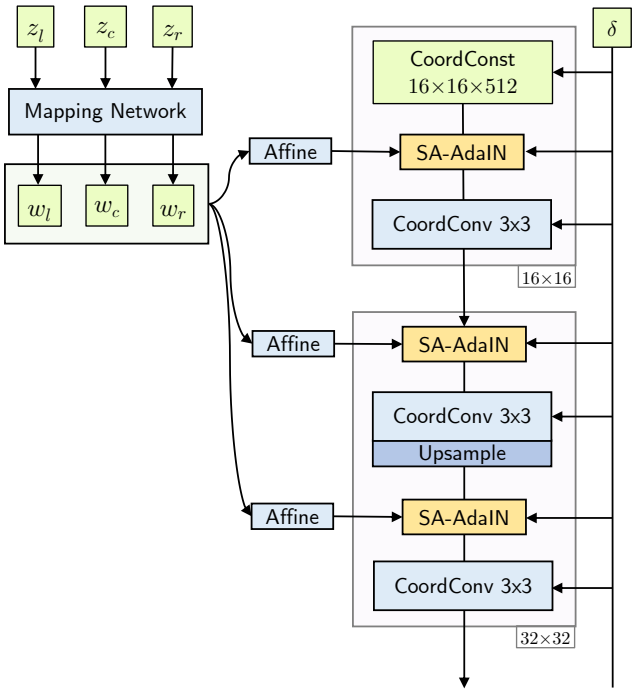


Figure 4: Illustration of our proposed generator. In this illustration, we omit some standard StyleGAN2 layers that are not essential for our architecture to not clutter the exposition. The full architecture is provided in Appendix B. CoordConst and CoordConv3x3 are analogs of StyleGAN2's Const and Conv3x3 blocks, but with coordinates embeddings concatenated to the hidden representations (the same way as in INR-GAN [43]).

3.1. Aligning Latent and Image Spaces

The core idea of our model is positioning *global* latent codes (*anchors*) on the image coordinates grid and slowly varying them with linear interpolation while moving along the plane. In this way, interpolation between the latent codes can be seen as the camera translation between the scenes corresponding to neighboring anchors. We call those anchors *global* because each one of them influences many frames. The idea for 1D alignment (i.e., where we move only along a single axis) is illustrated on Figure 2.

Imagine that we need to generate a pixel or intermediate feature value $\boldsymbol{v}$ at position $(x, y) \in \mathbb{R}^2$. A traditional generator would produce a latent code $\boldsymbol{w}$ and generate the value based on it: $\boldsymbol{v}(x, y) = \mathsf{G}(x, y; \boldsymbol{w})$. But in our case, we make the latent code be dependent on x (it's trivial to generalize it to be dependent on both x and y):

$$\boldsymbol{v}(x, y) = \mathsf{G}(x, y; \boldsymbol{w}_x), \tag{1}$$

where $\boldsymbol{w}_x$ is computed as a linear interpolation between the nearby anchors $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$ at positions $a, b \in \mathbb{R}$:

$$\boldsymbol{w}_x = \alpha \boldsymbol{w}_a + (1 - \alpha)\boldsymbol{w}_b, \tag{2}$$

and α is the normalized distance from x to b: $\alpha = (b - x)/(b - a)$.

In this way, latent and image spaces become aligned with one another: any movement in the coordinate space spurs movement in the latent space and vice versa. By training G to produce images in the interpolated regions, it implicitly learns to connect $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$ with realistic inter-scene frames. At test time, this allows us to connect any two independent $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$ into a single panorama by generating in-between frames, as can be seen in Figure 1.

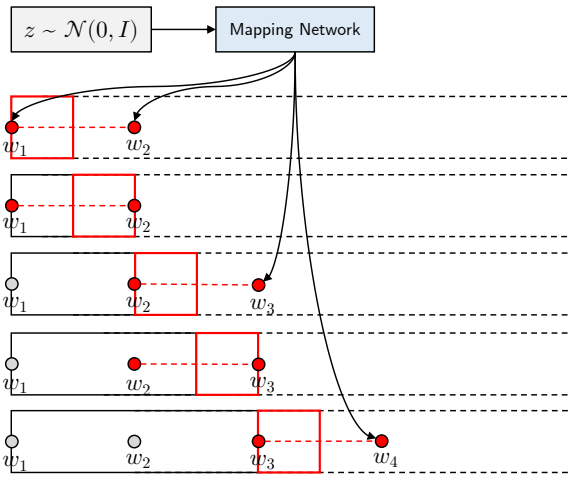


Figure 5: Inference process of our model at test time for $d = 2W$ (i.e. the distance between anchors is twice larger than the frame width). We sample new anchors $\boldsymbol{w}_i$ on the fly at positions $0, d, 2d, 3d, \ldots$. Since only relative positional information is provided to the decoder, we can decode images at any location $x \in (-\infty, \infty)$.

3.2. Spatially Aligned AdaIN (SA-AdaIN)

Our G architecture is based on StyleGAN2's generator, which uses an AdaIN-like mechanism of modulating the decoder's convolutional weights. This mechanism is not suited for our setup since one cannot make convolutional weights be dependent on the coordinate position efficiently. That is why we develop a specialized AdaIN variation that can be implemented efficiently on modern hardware.

AdaIN [18] works the following way: given input $\boldsymbol{h} \in \mathbb{R}^{c \times s^2}$ of resolution s^2 (for simplicity, we consider it to be square), it first normalizes it across spatial dimensions and then rescales and shifts with parameters $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^c$:

$$\text{AdaIN}(\boldsymbol{h}, \boldsymbol{\gamma}, \boldsymbol{\beta}) = \boldsymbol{\gamma} \cdot \frac{\boldsymbol{h} - \boldsymbol{\mu}(\boldsymbol{h})}{\boldsymbol{\sigma}(\boldsymbol{h})} + \boldsymbol{\beta} \tag{3}$$

where $\boldsymbol{\mu}(\boldsymbol{h}), \boldsymbol{\sigma}(\boldsymbol{h}) \in \mathbb{R}^c$ are mean and standard deviation computed across the spatial axes and all the operations are applied element-wise. It was shown that dropping the shifting operation does not affect the performance much [24]:

$$\text{AdaIN}'(\boldsymbol{h}, \boldsymbol{\gamma}) = \boldsymbol{\gamma} \cdot \boldsymbol{h} / \boldsymbol{\sigma}(\boldsymbol{h}) \tag{4}$$

thus we build on top of this simplified version of AdaIN.

Our Spatially-Aligned AdaIN (SA-AdaIN) is an analog of AdaIN' for a scenario where latent and image spaces are aligned with one other (as described in Sec 3.1), i.e. where the latent code is different depending on the coordinate position we compute it in. This section describes it for 1D-case, i.e., when the latent code changes only across the horizontal axis, but our exposition can be easily generalized to the 2D case (actually, for any N-D).

While generating an image, ALIS framework uses $\boldsymbol{w}_l, \boldsymbol{w}_c$ and $\boldsymbol{w}_r$ to compute the interpolations $\boldsymbol{w}_x$ for the required positions x. However, following StyleGAN2, we input to the convolutional layers not the "raw" latent codes $\boldsymbol{w}$, but style vectors $\boldsymbol{\gamma}$ obtained through an affine transform $\boldsymbol{\gamma} = \boldsymbol{A}^\ell \boldsymbol{w} + \boldsymbol{b}^\ell$ where ℓ denotes the layer index. Since a linear interpolation and an affine transform are interchangeable, from the performance considerations we first compute $\boldsymbol{\gamma}_l, \boldsymbol{\gamma}_c, \boldsymbol{\gamma}_r$ from $\boldsymbol{w}_l, \boldsymbol{w}_c, \boldsymbol{w}_r$ and then compute the interpolated style vectors $\boldsymbol{\gamma}_x$ instead of interpolating $\boldsymbol{w}_l, \boldsymbol{w}_c, \boldsymbol{w}_r$ into $\boldsymbol{w}_x$ immediately.

SA-AdaIN works the following way. Given anchor style vectors $\boldsymbol{\gamma}_l, \boldsymbol{\gamma}_c, \boldsymbol{\gamma}_r$, image offset δ, distance d between the anchors and the resolution of the hidden representation s, it first computes the grid of interpolated styles $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, ..., \boldsymbol{\gamma}_s] \in \mathbb{R}^{s \times c}$, where:

$$\boldsymbol{\gamma}_k = \begin{cases} \frac{d-\delta-k/s}{d}\boldsymbol{\gamma}_l + \frac{\delta+k/s}{d}\boldsymbol{\gamma}_c, & \text{if } \delta + k/s > d \\ \frac{2d-\delta-k/s}{d}\boldsymbol{\gamma}_c + \frac{\delta+k/s-d}{d}\boldsymbol{\gamma}_r, & \text{otherwise} \end{cases} \tag{5}$$

This formulation assumes that anchors $\boldsymbol{\gamma}_l, \boldsymbol{\gamma}_c, \boldsymbol{\gamma}_r$ are located at positions $-d, 0, d$ respectively.

Then, just like AdaIN', it normalizes $\boldsymbol{h}$ to obtain $\tilde{\boldsymbol{h}} = \boldsymbol{h}/\boldsymbol{\sigma}(\boldsymbol{h})$. After that, it element-wise multiplies $\boldsymbol{\Gamma}$ and $\boldsymbol{h}$, broadcasting the values along the vertical positions:

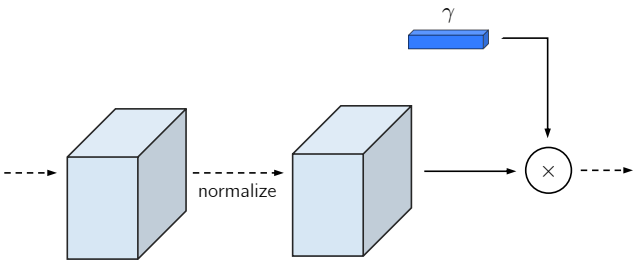
$$[\text{SA-AdaIN}(\boldsymbol{x}, \boldsymbol{\gamma}_l, \boldsymbol{\gamma}_c, \boldsymbol{\gamma}_r, \delta)]_k = \boldsymbol{\gamma}_k \cdot [\boldsymbol{h}/\boldsymbol{\sigma}(\boldsymbol{h})]_k, \tag{6}$$

where we denote by $[\cdot]_k$ the k-th vertical patch of a variable of size $s \times 1 \times c$. Note that since our G produces images in a patch-wise fashion, we normalize across patches instead of the full images. Otherwise, it will break the equivariance and lead to seams between nearby frames.

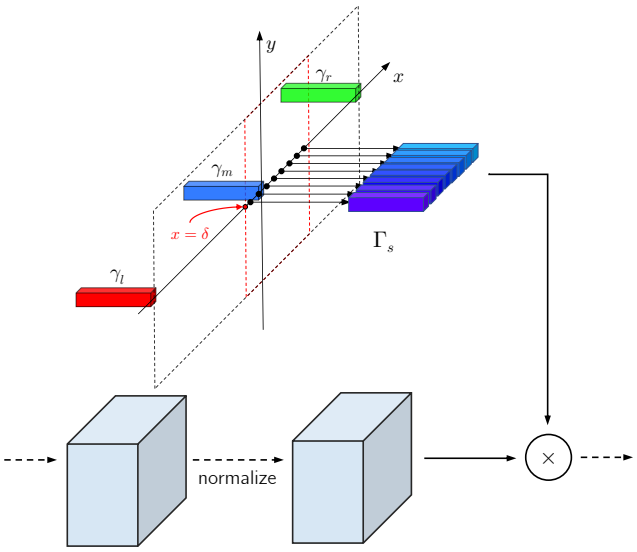
SA-AdaIN is illustrated in Figure 6.

3.3. Which datasets are "connectable"?

As mentioned in [10], to generate arbitrarily-sized images, we want the data statistics to be invariant to their spatial location in an image. This means that given an image



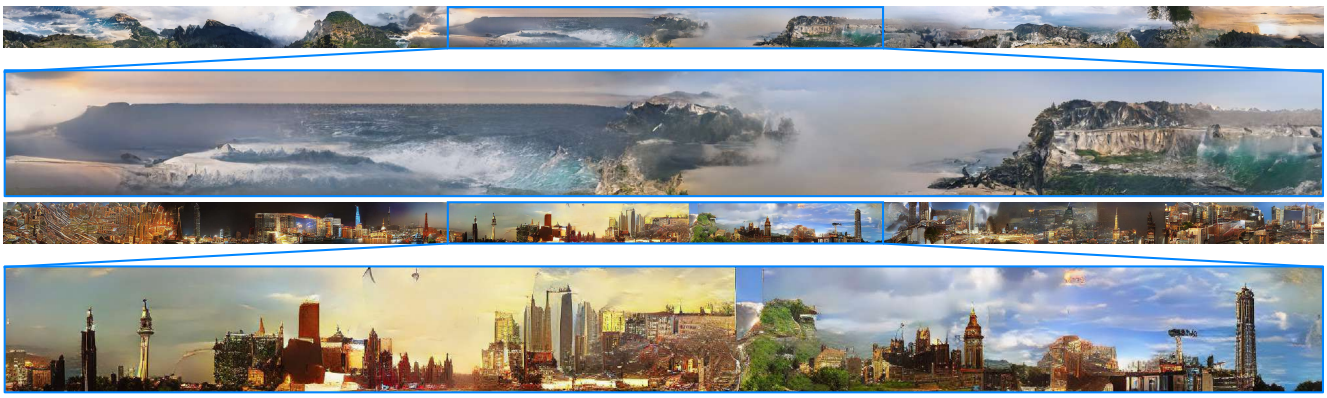
(a) AdaIN without shifting [24]



(b) Spatially-Aligned AdaIN

Figure 6: Top: AdaIN [18] without shifting (as explored in [24]). Bottom: Spatially-Aligned AdaIN (SA-AdaIN). Style vectors $\boldsymbol{\gamma}_\ell, \boldsymbol{\gamma}_c, \boldsymbol{\gamma}_r$ are positioned on the 2D coordinate grid and a style vector in each location is computed as a linear interpolation between neighboring anchors. For both AdaIN and SA-AdaIN, we broadcast styles' dimensions in the multiplication operation to match the input tensor shape.

patch, one should be unable to confidently predict which part of an image it comes from. However, many datasets either do not have this property (like FFHQ [23]) or have it only for a small number of images. To check if images in a given dataset have spatially invariant statistics and to extract a subset of such images, we developed the following simple procedure. Given a dataset, we train a classifier on its patches to predict what part of an image a patch is coming from (we allocate $< 10\%$ of the dataset to do this). If a classifier cannot do this easily (i.e., the accuracy is low), then the dataset does have spatially invariant statistics. To extract a subset of images with spatially invariant statistics, we measure the classifier's confidence on each image and select those for which its confidence is low. The details are in Appendix C.



(a) Taming Transformers [10] for *unconditional* generation (the original paper mainly focused on the conditional generation from semantic masks/depth maps/etc).

(b) LocoGAN [57] in the StyleGAN2 [24] framework + Fourier positional embeddings [42, 45]. The method generates realistic scenes, but has repeated content.

(c) ALIS (ours). The method generates diverse infinite scenes without repetition or stitching artifacts.

Figure 7: Qualitative comparison between different methods on LHQ and LSUN Tower. More samples are in Appendix F.

4. Landscapes HQ dataset

We introduce Landscapes HQ (LHQ): a dataset of 90k high-resolution ($\geq 1024^2$) nature landscapes that we crawled and preprocessed from two sources: Unsplash (60k) and Flickr (30k). We downloaded 500k of images in total using a list of manually constructed 450 search queries and then filtered it out using a blacklist of 320 image tags. After that, we ran a pretrained Mask R-CNN to remove the pictures that likely contain objects on them. As a result, we obtained a dataset of 90k high-resolution images. Finally, we *manually* filtered out 10k images to construct LHQ-base: a landscapes dataset that is guaranteed to include only high-quality images. The details are described

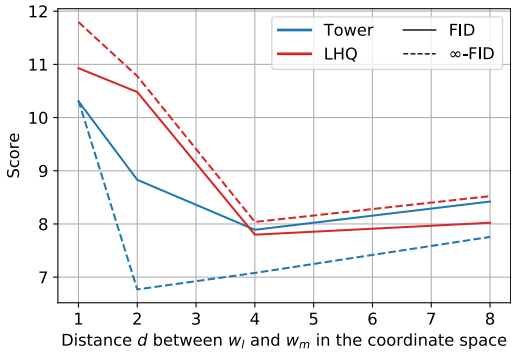


Figure 8: Varying distance d between the anchors for LSUN Tower and LHQ datasets. Larger distance leads to better per-frame image quality, but produces repetitions artifacts as depicted on Figure 10.

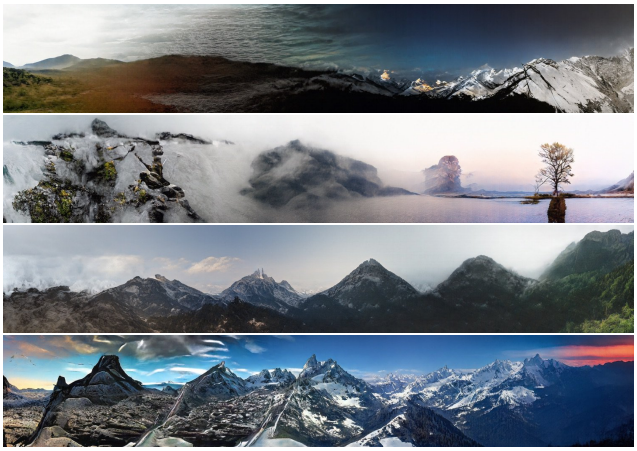


Figure 9: Failure cases of our method: 1) top 2 rows — connecting too different anchors (like close-by water and far-away mountains); and 2) bottom 2 rows — content repetitions (which arise due to the usage of periodic positional embeddings [45, 42]).

in Appendix D.

5. Experiments

Datasets. We test our model on 4 datasets: LSUN Tower 256^2, LSUN Bridge 256^2 and LHQ 256^2 (see Sec 4). We preprocessed each dataset with the procedure described in Algorithm 1 in Appendix C to extract a subset of data with (approximately) spatially invariant statistics. We emphasize that we focus on infinite horizontal generation, but the framework is easily generalizable for the joint vertical+horizontal infinite generation. We also train ALIS on LHQ 1024^2 to test how it scales to high resolutions.

Evaluation. We use two metrics to compare the methods: a popular FID measure [17] and our additionally proposed ∞-FID, which is used to measure the quality and diversity of an "infinite" image. It is computed in the following way. For a model with the output frame size of 256^2, we first generate a very wide image of size $256 \times (50000 \cdot 256)$. Then, we slice it into 50000 frames of size 256^2 without overlaps or gaps and compute FID between the real dataset and those sliced frames. We also compute a total number of parameters and inference speed of the generator module for each baseline. Inference speed is computed as a speed to generate a *single* frame of size 256^2 on NVidia V100 GPU.

Figure 10: A problem of using large distance d between the anchors (in the above case, model was trained with $d = 16$). Though it improves per-frame image quality, the model starts repeating itself during the generation process. For all the datasets, we use the preprocessing procedure described in Sec 3.3.

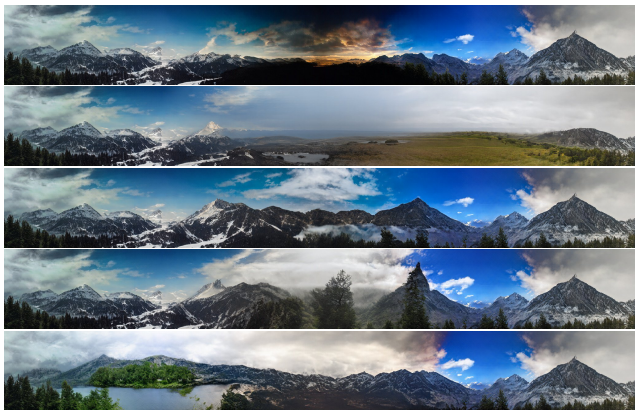


Figure 11: ALIS allows to resample any part of an image without breaking its "connectivity": frames are still consistent both locally and globally.

Baselines. For our main baselines, we use two methods: Taming Transformers (TT) [10] and LocoGAN [57]. For LocoGAN, since by default it is a small model (5.5M parameters vs ≈50M parameters in StyleGAN2) and does not employ any StyleGAN2 tricks that boost the performance (like style mixing, equalized learning rate, skip/resnet connections, noise injection, normalizations, etc.) we reimplemented it on top of the StyleGAN2 code to make the comparison fair. We also replaced raw coordinates conditioning with Fourier positional embeddings since raw coordinates do not work for $(-\infty, \infty)$ range and were recently shown to be inferior [42, 45, 43, 1]. We called this model LocoGAN+SG2+Fourier. Besides the methods for infinite image

Table 1: Scores for different models on different datasets in terms of FID and ∞-FID. "N/A" denotes "not-applicable".

Method	Bridge 256^2		Tower 256^2		Landscapes 256^2		Speed	#params
	FID	∞-FID	FID	∞-FID	FID	∞-FID		
Taming Transformers [10]	56.06	58.27	50.16	51.32	61.95	64.3	9981 ms/img	377M
LocoGAN [57]+SG2+Fourier	9.02	264.7	8.36	381.1	7.82	211.2	74.7 ms/img	53.7M
ALIS (ours)	10.24	10.79	8.83	8.99	10.48	10.64	53.9 ms/img	48.3M
w/o coordinates	13.21	13.92	10.32	10.17	12.63	13.07	46.8 ms/img	47.1M
StyleGAN2 (config-e)	7.33	N/A	6.75	N/A	3.94	N/A	32.4 ms/img	47.1M

generation, we compute the performance of the traditional StyleGAN2 as a lower bound on the possible image generation quality on a given dataset.

Each model was trained on 4 v100 GPUs for 2.5 days, except for TT, which was trained for 5 days since it is a two-stage model: 2.5 days for VQGAN and then 2.5 days for Transformer. For TT, we used the official implementation with the default hyperparameters setup for unconditional generation training. See Appendix E for the detailed remarks on the comparison to TT. For StyleGAN2-based models, we used config-e setup (i.e. half-sized high-resolution layers) from the original paper [24]. We used precisely the same training settings (loss terms, optimizer parameters, etc.) as the original StyleGAN2 model.

Ablations. We also ablate the model in terms of how vital the coordinates information is and how distance between anchors influences generation. For the first part, we replace all Coord-* modules with their non-coord-conditioned counterparts. For the second kind of ablations, we vary the distance between the anchors in the coordinate space for values $d = 1, 2, 4, 8$. This distance can also be understood as an aspect ratio of a single scene.

Results. The main results are presented in Table 1, and we provide the qualitative comparison on Figure 7. To measure ∞-FID for TT, we had to simplify the procedure since it relies on GroupNorm in its decoder and generated 500 images of width 256×100 instead of a single image of width 256×50000. We emphasize, that it is an *easier* setup and elaborate on this in Appendix E. Note also, that the TT paper [10] mainly focused on *conditional* image generation from semantic masks/depths maps/class information/etc, that's why the visual quality of TT's samples is lower for our *unconditional* setup.

The ∞-FID scores on Table 1 demonstrate that our proposed approach achieves state-of-the-art performance in the generation of infinite images. For the traditional FID measured on independent frames, its performance is on par with LocoGAN. However, the latter completely diverges in terms of infinite image generation because spatial noise does not provide global variability, which is needed to generate diverse scenes. Moreover, we noticed that LocoGAN has learned to ignore spatial noise injection [23] to make it easier for the decoder to stitch nearby patches, which shuts off its only source of scene variability. For the model without coordinates conditioning, image quality drops: visually, they become blurry (see Appendix F) since it is harder for the model to distinguish small shifts in the coordinate space. When increasing the coordinate distance between anchors (the width of an image equals 1 coordinate unit), traditional FID improves. However, this leads to repetitious generation, as illustrated in Figure 10. It happens due to short periods of the periodic coordinate embeddings and anchors changing too slowly in the latent space. When most of the positional embeddings complete their cycle, the model has not received enough update from the anchors' movement and thus starts repeating its generation. The model trained on 1024^2 crops of LHQ achieves FID/∞-FID scores of 10.11/10.53 respectively and we illustrate its samples in Figure 1 and Appendix F.

As depicted in Figure 9, our model has two failure modes: sampling of *too* unrelated anchors and repetitious generation. The first issue could be alleviated at test-time by using different sampling schemes like truncation trick [3, 24] or clustering the latent codes. A more principled approach would be to combine autoregressive inference of TT [10] with our ideas, which we leave for future work.

One of our model's exciting properties is the ability to replace or swap any two parts of an image without breaking neither its local nor global consistency. It is illustrated in Figure 11 where we resample different regions of the same landscape. For it, the middle parts are changing, while the corner ones remain the same.

6. Conclusion

In this work, we proposed an idea of aligning the latent and image spaces and employed it to build a state-of-the-art model for infinite image generation. We additionally proposed a helpful ∞-FID metric and a simple procedure to extract from any dataset a subset of images with approximately spatially invariant statistics. Finally, we introduced LHQ: a novel computer vision dataset consisting of 90k high-resolution nature landscapes.

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. *arXiv preprint arXiv:2011.13775*, 2020. 2, 7, 13, 14

[2] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. *arXiv preprint arXiv:1705.06566*, 2017. 1, 2, 12, 13, 14

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 8

[4] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *arXiv preprint arXiv:2012.00926*, 2020. 2

[5] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. *arXiv preprint arXiv:2012.09161*, 2020. 2

[6] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 2

[7] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2019. 3

[8] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018. 3

[9] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016. 3

[10] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020. 1, 2, 3, 5, 6, 7, 8, 11, 12, 13, 14, 17, 19, 20

[11] Anna Frühstück, Ibraheem Alhashim, and Peter Wonka. Tilegan: synthesis of large-scale non-homogeneous textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–11, 2019. 1, 2, 13

[12] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 3

[13] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017. 2

[14] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 2

[15] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017. 2

[16] Dongsheng Guo, Hongzhi Liu, Haoru Zhao, Yunhao Cheng, Qingwei Song, Zhaorui Gu, Haiyong Zheng, and Bing Zheng. Spiral generative network for image extrapolation. In *European Conference on Computer Vision*, pages 701–717. Springer, 2020. 3

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017. 7

[18] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 2, 3, 4, 5

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 18

[20] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 1, 2, 12, 13, 14

[21] Biliana Kaneva, Josef Sivic, Antonio Torralba, Shai Avidan, and William T Freeman. Infinite images: Creating and exploring a large photorealistic virtual space. *Proceedings of the IEEE*, 98(8):1391–1407, 2010. 3

[22] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020. 13

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 3, 5, 8

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 2, 3, 5, 6, 8, 13

[25] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. COCO-GAN: generation by parts via conditional coordinating. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 3, 12, 13, 14

[26] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1824–1833, 2019. 2

[27] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. *arXiv preprint arXiv:2012.09855*, 2020. 3

[28] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*, 2018. 2

[29] Chaochao Lu, Richard E. Turner, Yingzhen Li, and Nate Kushman. Interpreting spatially infinite generative models, 2020. 1, 2, 12, 13, 14

[30] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020. 2

[31] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. 13

[32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 2

[33] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 2

[34] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers*, pages 614–623, 2006. 2

[35] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. *arXiv preprint arXiv:2011.12100*, 2020. 2

[36] Chris Olah, Nick Cammarata, Chelsea Voss, Ludwig Schubert, and Gabriel Goh. Naturally occurring equivariance in neural networks. *Distill*, 2020. https://distill.pub/2020/circuits/equivariance. 3

[37] Yoav IH Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308, 2001. 2

[38] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 2

[39] Tiziano Portenier, Siavash Bigdeli, and Orçun Göksel. Gramgan: Deep 3d texture synthesis from 2d exemplars. *arXiv preprint arXiv:2006.16112*, 2020. 2

[40] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Computer Vision (ICCV), IEEE International Conference on*, 2019. 1, 2

[41] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020. 2

[42] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 2, 3, 6, 7, 12, 15

[43] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. *arXiv preprint arXiv:2011.12026*, 2020. 1, 2, 3, 4, 7, 12, 13, 14

[44] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006. 3

[45] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 2, 3, 6, 7, 12, 15

[46] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10521–10530, 2019. 3

[47] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 3

[48] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 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*, pages 6924–6932, 2017. 3

[49] Basile Van Hoorick. Image outpainting and harmonization using generative adversarial networks. *arXiv preprint arXiv:1912.10960*, 2019. 3

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 2

[51] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1399–1408, 2019. 3

[52] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 18, 19

[53] Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in gans. *arXiv preprint arXiv:2012.05217*, 2020. 1

[54] Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10561–10570, 2019. 3

[55] Richard Zhang. Making convolutional networks shift-invariant again. In *International Conference on Machine Learning*, pages 7324–7334. PMLR, 2019. 3, 11, 12

[56] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*, 2020. 13

[57] Łukasz Struski, Szymon Knop, Jacek Tabor, Wiktor Daniec, and Przemysław Spurek. Locogan – locally convolutional gan, 2020. 1, 2, 3, 6, 7, 8, 12, 13, 14