# InfinityGAN: Towards Infinite-Resolution Image Synthesis

Chieh Hubert Lin[1], Hsin-Ying Lee[2], Yen-Chi Cheng[3], Sergey Tulyakov[2], Ming-Hsuan Yang[1,4]

[1]UC Merced, [2]Snap Inc., [3]CMU, [4]Google Research

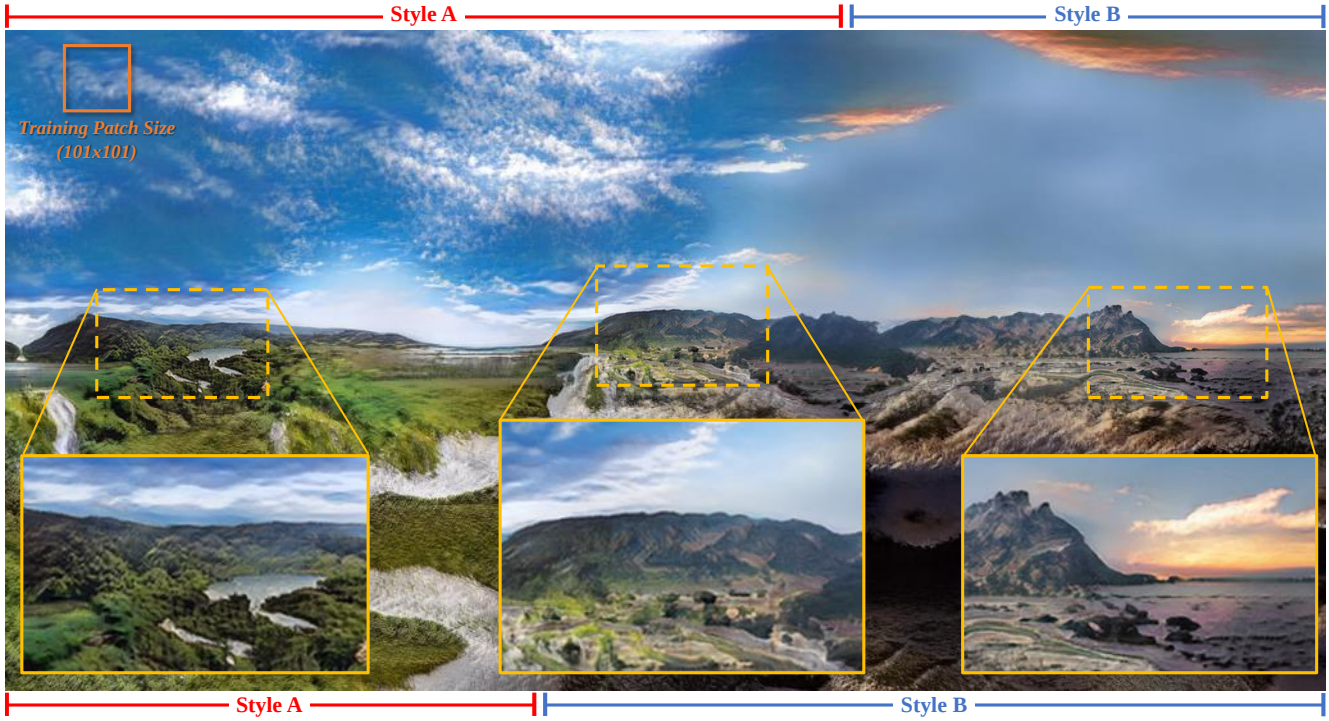


Figure 1: **Synthesizing infinite-resolution images from finite-resolution inputs.** A 1024×2048 image composed of 242 patches, independently synthesized by InfinityGAN with spatial fusion of two styles. The generator is trained on 101×101 patches (e.g., marked in top-left) sampled from 197×197 real images. Note that training and inference (of any resolution) are performed on a single GTX TITAN X GPU. Zoom-in for better experience.

## Abstract

*We present InfinityGAN, a method to generate arbitrary-resolution images. The problem is associated with several key challenges. First, scaling existing models to a high resolution is resource-constrained, both in terms of computation and availability of high-resolution training data. InfinityGAN trains and infers patch-by-patch seamlessly with low computational resources. Second, large images should be locally and globally consistent, avoid repetitive patterns, and look realistic. To address these, InfinityGAN takes global appearance, local structure and texture into account. With this formulation, we can generate images with resolution and level of detail not attainable before. Experimental evaluation supports that InfinityGAN generates images with superior global structure compared to baselines at the same time featuring parallelizable inference. Finally, we show several applications unlocked by our approach, such as fusing styles spatially, multi-modal outpainting and image inbetweening at arbitrary input and output resolutions.*

## 1. Introduction

*"To infinity and beyond!"* – *Buzz Lightyear*

Generative models [7] witness substantial improvements in resolution and level of details. Most improvements come at a price of increased training time [8, 23], larger model size [2], and stricter data requirements [12]. The most recent works synthesize images at 1024×1024 resolution featuring a high level of details and fidelity. However, straightforward scaling of such models to higher resolutions is problematic.

Project Page: http://bit.ly/InfinityGANProject

Synthesizing infinite-resolution images is constrained by the finite nature of resources. Finite computational resources (e.g. memory and training time) set bounds for input receptive field and output resolution. A further limitation is that there exists no infinite resolution image dataset. Thus, to generate infinite resolution images, a model should learn the implicit global structure without direct supervision and under limited computational resources.

Repetitive texture synthesis methods [5, 39] generalize to high resolutions. Yet, such methods are not able to synthesize real-world images. Recent works, such as SinGAN [32] and InGAN [33], learn an internal patch distribution for image synthesis. Although these models can generate images with arbitrary shapes, in a nutshell, both approaches memorize objects and patterns of the scene with position encoding [11], then reach visual diversity by permuting and randomizing the repetitive texture. In Section A, we show that these models do not infer structural relationships well, and fail to construct plausible holistic views with spatially extended latent space. A different approach, COCO-GAN [17], learns a coordinate-conditioned patch distribution for image synthesis. Despite the ability to slightly extend images beyond the learned boundary, it fails to maintain the global coherence of the generated images when scaling to a $2\times$ higher resolution.

*How to generate infinite-resolution images?* Humans are able to guess the whole scene given a partial observation of it. In a similar fashion, we aim to build a generator that trains with image patches, and at inference time synthesizes images well beyond its training data resolution. The generator can thus generalize to an unbounded arbitrarily-high resolution. An example of a synthesized scene containing heterogeneous and diverse textures is shown in Figure 1.

We propose InfinityGAN, a method that trains on a finite and low-resolution dataset, while generating infinite-resolution images at inference time. To reach global consistency, we assume that all patches in a large image share the same holistic appearance. Local texture and structure are modeled separately, allowing the method to synthesize diverse local details. InfinityGAN consists of a neural implicit function, termed *structure synthesizer*, and a fully-convolutional StyleGAN2 generator, dubbed *texture synthesizer*. The structure synthesizer conditions on global appearance, samples a sub-region from an implicit image that is extendable to an infinite resolution, and produces local structural representations. The texture synthesizer is designed to avoid common position encoding of CNNs [11] and generates texture for the structure provided by the structure synthesizer. Consisting of these two modules, InfinityGAN can infer a compelling global composition of a scene and realistically render its local details. Trained on small patches, InfinityGAN achieves high-quality, seamless and high-resolution outputs, requiring low computational resources—a single TITAN X to train and test.

We conduct extensive experiments to validate the proposed method. Qualitatively, we present the everlastingly long landscape images. Quantitatively, we evaluate InfinityGAN and related methods using user study and a proposed ScaleInv FID metric. Furthermore, we demonstrate the efficiency and efficacy of the proposed methods with several applications. First, we demonstrate the flexibility and controllability of the proposed method by spatially fusing structures and textures from different distributions within an image. Second, we show that our model is an effective deep image prior for the image outpainting task with the image inversion technique and achieves multi-modal outpainting of arbitrary length from arbitrarily-shaped inputs. Third, with the proposed model we can divide-and-conquer the full image generation into independent patch generation and achieve $7.2\times$ of inference speed-up with parallel computing, which is critical for high-resolution image synthesis.

## 2. Related Work

**Latent generative models.** Existing generative models are mostly designed to synthesize images of fixed-resolution. A few methods [12, 13, 29] have been recently developed to train latent generative models on high-resolution images, up to $1024\times1024$ pixels. However, latent generative models generate images from dense latent vectors and require synthesizing all structural contents at once. Bounded by computational resources, these approaches can synthesize images of certain resolutions. In contrast, GANs trained based on image patches [17, 32, 33] are less constrained by the resource bottleneck with the synthesis-by-part approach. However, [32, 33] can only model and repeat intrinsic statistics of a single image, and [17] extrapolate a few patches from the image center.

**Conditional generative models.** Numerous vision tasks such as image super-resolution, semantic image synthesis and image extrapolation often showcase high-quality results over $1024\times1024$ pixels. These tasks are less related to our setting, since most structural information is already provided in the conditional inputs. Image super-resolution methods [15, 16] typically copy neighbor pixels from the conditional inputs, and do not synthesize novel structural contents or objects. Semantic image synthesis [6, 28, 37] approaches leverage the input semantic segmentation map that contains the structural specification and generate correlated results by filling categorical content.Image outpainting [1, 18, 30, 40] is related to image inpainting [19, 41] and shares similar issues that the generator tends to copy-and-paraphrase the conditional input or create mottled textural samples, leading to repetitive results especially when the outpainted region is large. InOut [4] proposes to outpaint image with GANs inversion and yield results with higher diversity. We show that with InfinityGAN as the deep image prior along with [4], we obtain the state-of-the-art outpaint-

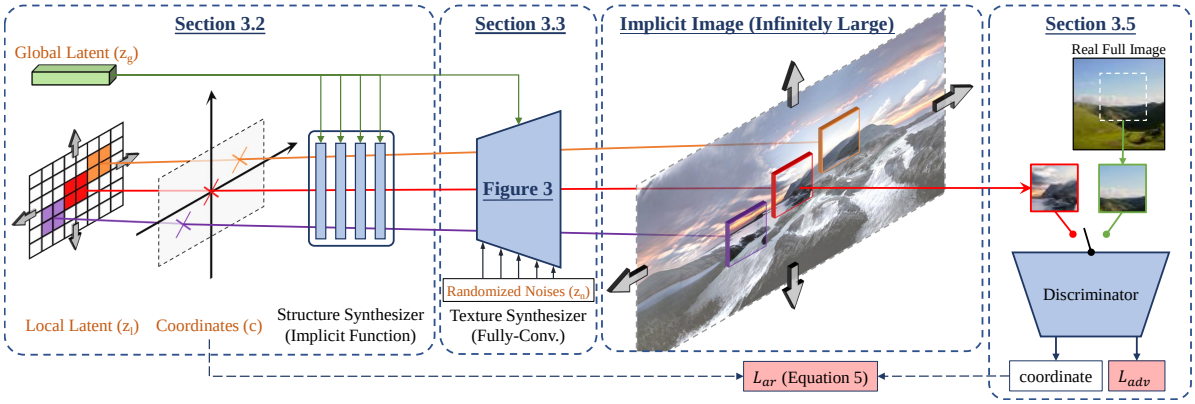


Figure 2: **Overview.** The generator of InfinityGAN consists of two modules, a structure synthesizer based on a neural implicit function, and a fully-convolutional texture synthesizer with all positional information removed (see Figure 3). The two networks take four sets of inputs, a global latent variable that defines the holistic appearance of the image, a local latent variable that represents the local and structural variation, a continuous coordinate for learning the neural implicit structure synthesizer, and a set of randomized noises to model fine-grained texture. InfinityGAN synthesizes images of arbitrary resolution by learning spatially extensible representations.

ing results and avoids the need of sequential outpainting. We illustrate and compare the characteristics of these tasks against our setting in the supplementary material.

**Neural implicit representation.** Neural implicit functions [27, 24, 25] have been applied to model the structural information of 3D and continuous representations. Similar to methods in spirit, our query-by-coordinate synthesizer is able to model structural information effectively.

## 3. Proposed Method

An arbitrarily-large image can be described globally and locally. Globally, images should be coherent and, hence, global characteristics should be expressible by a relatively compact *holistic appearance*. It allows humans to express image content by giving brief descriptions (e.g. a medieval landscape, ocean view panorama). Hence, we assume that holistic appearance is fixed for each image and represents the high-level composition and content of the scene.

Locally, a close-up of an image is defined by its local and neighboring structure and texture. The structural content represents objects, shapes and their arrangement within a local region. Once the structure is defined, there exist multiple feasible textures to render realistic scenes. The realism of each synthesized texture implicitly considers the material or lighting of the objects it attains to in the scene. Hence, texture synthesis can be seen as a second step, conditioned on the structure, material properties and lighting.

Despite being local, structure and texture should conform to the global *holistic appearance* such that the structure of nearby objects is consistent with their texture-related properties shared among the neighboring patches. Based on these considerations, an infinite-resolution image can be generated. Once the global, holistic appearance is sampled, local structure and texture can be spatially extended infinitely as long as they follow holistic appearance.

### A. Overview

We aim to synthesize infinite-resolution images using only low-resolution patches during training. Based on the above analysis, InfinityGAN consists of two components: a *structure synthesizer* $G_{\mathrm{S}}$ implemented as an implicit function modeling the holistic appearance of an image, and a *texture synthesizer* $G_{\mathrm{T}}$ implemented as a fully-convolutional StyleGAN2 [13] modeling the local texture properties. Figure 2 presents an overview of our framework.

Four latent variables control the image generation process. A global latent variable $\mathbf{z}_{\mathrm{g}}$ is provided to both $G_{\mathrm{S}}$ and $G_{\mathrm{T}}$ to enforce that the holistic appearance of the whole image is taken into account for each patch. $G_{\mathrm{S}}$ renders structure for each patch located as specified by the coordinate grid $\mathbf{c}$. Local variation of patches is modeled using the local latent code $\mathbf{z}_{\mathrm{l}}$. Since there exist multiple possible textures once the structure is defined, each layer of $G_{\mathrm{T}}$ receives additional conditioning $\mathbf{z}_{\mathrm{n}}$, to model local fine-grained details absent in $\mathbf{z}_{\mathrm{g}}$. Denoting the generated patch at location $\mathbf{c}$ as $\mathbf{p}_{\mathrm{c}}$ the generative process writes as:

$$\mathbf{z}_{\mathrm{S}} = G_{\mathrm{S}}(\mathbf{z}_{\mathrm{g}}, \mathbf{z}_{\mathrm{l}}, \mathbf{c}), \mathbf{p}_{\mathrm{c}} = G_{\mathrm{T}}(\mathbf{z}_{\mathrm{S}}, \mathbf{z}_{\mathrm{g}}, \mathbf{z}_{\mathrm{n}}), \qquad (1)$$

with $\mathbf{z}_{\mathrm{S}}$ denoting a structure latent variable. We detail each part in the following sections.

### B. Structure Synthesizer ($G_{\mathrm{S}}$)

Structure synthesizer is an implicit function implemented with a neural network. Here, the goal of the neural implicit function is to sample an implicit representation conditioned on the global $\mathbf{z}_{\mathrm{g}}$ and local $\mathbf{z}_{\mathrm{l}}$ latent variables, and generate the structure at the queried location $\mathbf{c}$.

The global latent variable $\mathbf{z}_{\mathrm{g}}$ serves as a global appearance descriptor. We sample $\mathbf{z}_{\mathrm{g}}$ from a unit Gaussian distribution once and inject $\mathbf{z}_{\mathrm{g}}$ into every layer and every pixel in $G_{\mathrm{S}}$ via feature modulation [10]. Local variations of image structure are represented by $\mathbf{z}_{\mathrm{l}}$. As local variations are

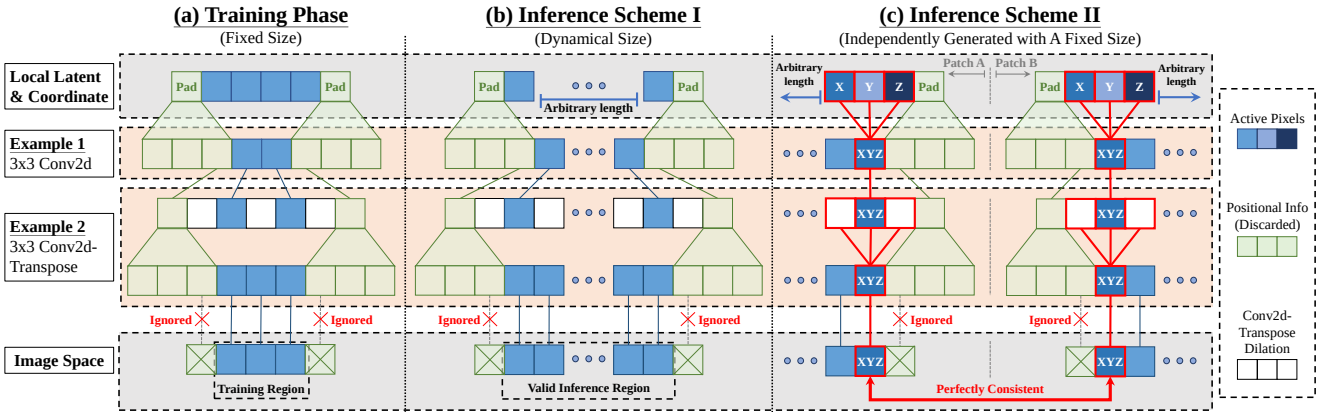


Figure 3: **Fully-convolutional generator without paddings.** To make the texture synthesizer learn only texture representations rather than structural information, we remove all paddings from the generator to avoid position encoding. With such a fully-convolutional generator design, we can spatial-independently generate patches and forming into a seamless image.

positionally independent across the spatial dimension, we independently sample them from a unit Gaussian prior for each spatial position of $\mathbf{z}_{\mathrm{l}}$, forming a volumetric tensor that infinitely extensible in the spatial dimension. We provide $\mathbf{z}_{\mathrm{l}}$ as the input to $G_{\mathrm{S}}$. Note that using spatially-dependent variables as inputs in a neural implicit function is less common, but similar formulations can be found in [3, 31].

Finally, with the sampled implicit representation conditioned on $\mathbf{z}_{\mathrm{g}}$ and arbitrarily large $\mathbf{z}_{\mathrm{l}}$, the coordinate $\mathbf{c}$ serves as a query to obtain the region to be retrieved from the implicit image. To enable $G_{\mathrm{S}}$ to accept coordinates infinitely far we introduce a prior by exploiting the nature of landscape images: (a) self-similarity on the horizontal direction, and (b) rapid saturation to a single modal (e.g., land, sky or ocean) on the vertical direction. Accordingly, we use the periodic coordinates on the horizontal axis similar to [36], and project coordinates with a tanh function in the vertical direction. Furthermore, we use both sine and cosine functions to encode a single coordinate for numerical stability in the horizontal direction. Given horizontal and vertical indexes $(i_x, i_y)$ of $\mathbf{z}_{\mathrm{l}}$, we encode them as

$$\mathbf{c} = (\tanh(i_y), \cos(i_x/T), \sin(i_x/T)) \,, \tag{2}$$

where $T$ is the period of the sinusoid coordinates.

While coordinates repeat periodically, the variations of $\mathbf{z}_{\mathrm{l}}$ avoid repeating structures. To prevent the model from ignoring $\mathbf{z}_{\mathrm{l}}$ and generating repetitive content periodically in the horizontal direction, we adopt a mode-seeking diversity loss [22] between a pair of local latent variables $\mathbf{z}_{\mathrm{l}_1}$ and $\mathbf{z}_{\mathrm{l}_2}$:

$$\mathcal{L}_{\mathrm{div}} = \frac{\|G_{\mathrm{S}}(\mathbf{z}_{\mathrm{g}}, \mathbf{z}_{\mathrm{l}_1}, \mathbf{c}) - G_{\mathrm{S}}(\mathbf{z}_{\mathrm{g}}, \mathbf{z}_{\mathrm{l}_2}, \mathbf{c})\|_1}{\|\mathbf{z}_{\mathrm{l}_1} - \mathbf{z}_{\mathrm{l}_2}\|_1} \,. \tag{3}$$

Notice that $\mathbf{z}_{\mathrm{g}}$ and $\mathbf{c}$ are shared between $\mathbf{z}_{\mathrm{l}_1}$ and $\mathbf{z}_{\mathrm{l}_2}$.

Similar to [3], we adopt the feature unfolding technique to enable $G_{\mathrm{S}}$ to consider the information from a broader area of $\mathbf{z}_{\mathrm{l}}$ and $\mathbf{c}$. Given an intermediate feature $f$ in $G_{\mathrm{S}}$, the resulting feature map $u$ with $k \times k$ feature unfolding is:

$$u_{(i,j)} = \mathrm{Concat}(\{f(i+n, j+m)\}_{n,m \in \{-k/2, k/2\}}) \,, \tag{4}$$

where "Concat(·)" concatenates the unfolded vectors in the channel dimension. The use of feature unfolding turns $\mathbf{c}$ into a grid of coordinates rather than a simple triplet. In practice, as the grid-shaped $\mathbf{z}_{\mathrm{l}}$ and $\mathbf{c}$ are sampled with equal spacing between consecutive pixels, the feature unfolding can be efficiently implemented with CoordConv [20].

## C. Texture Synthesizer ($G_{\mathrm{T}}$)

The texture synthesizer is built upon the StyleGAN2 model [13]. First, we replace the fixed constant input with $\mathbf{z}_{\mathrm{S}}$, and use the original design of $\mathbf{z}_{\mathrm{n}}$ that injects randomized noises to model fine-grained random texture. Then, we use $\mathbf{z}_{\mathrm{g}}$ as the input to the mapping layer, which projects the single $\mathbf{z}_{\mathrm{g}}$ to layer-wise styles $\mathbf{z}_{\mathrm{T}}$ with multi-layer perceptrons. The styles $\mathbf{z}_{\mathrm{T}}$ are then injected into all pixels in each layer via feature modulation. Finally, we remove all zero paddings from the generator, as shown in Figure 3(a).

We remove all zero paddings for three major reasons. First, we observe that the StyleGAN2 model heavily relies on the position encoding [11] of CNNs through the zero paddings in the generator and memorizes the structural information from the training images. In Section B we show that StyleGAN2 fails to synthesize realistic images after all paddings have been removed from the generator. Second, the position encoding with zero paddings is a critical issue for generalizing the model to synthesize arbitrarily-large images with arbitrarily-large input latent variables. The third column of Figure 4 shows when we extend the input latent variable of the StyleGAN2 generator by multiple times, the center part of the features do not receive expected coordinate information from the paddings, resulting in extensively repetitive textures in the center area of the output image. Finally, the existence of paddings hampers $G$ from generating separate patches that can be composed together. Therefore, in order to take out all positional information from the generator, we remove all paddings, fa-

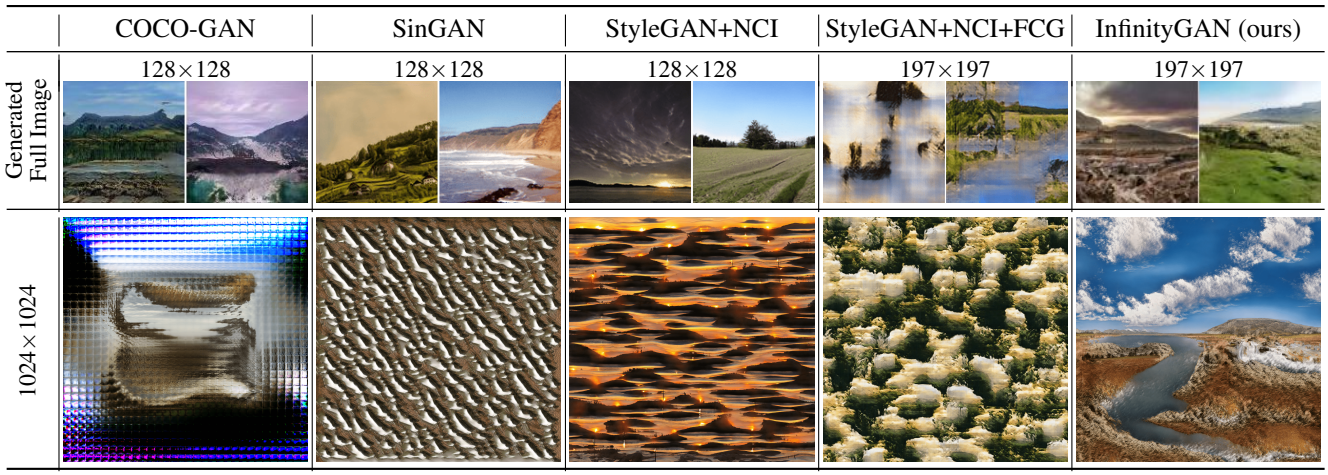


Figure 4: **Qualitative comparison.** We show that InfinityGAN can produce more favorable holistic appearances against related methods while testing with an extended resolution 1024×1024. (NCI: Non-Constant Input, FCG: Fully-Convolutional Generator). More results are shown in the supplementary materials.

cilitating the synthesis of arbitrary-resolution images. This way we enforce the model to fully rely on the structural features provided by $G_{\mathrm{S}}$, making $G_{\mathrm{T}}$ responsible for modeling texture-related details only. We refer to the proposed $G_{\mathrm{T}}$ as a fully-convolutional generator (FCG).

## D. Spatially Independent Generation

As shown in Figure 3(b), the removal of all paddings makes $G_{\mathrm{T}}$ a fully convolutional generator that synthesizes consistent outputs with arbitrarily sized $\mathbf{z}_{\mathrm{S}}$. Meanwhile, the $\mathbf{z}_{\mathrm{S}}$ generated with $G_{\mathrm{S}}$ is aligned to $\mathbf{z}_{\mathrm{l}}$, due to the nature of neural implicit functions. With these properties, we can form a one-to-one mapping from each pixel in the generated images, to a set of input latent variables that contribute to that pixel. That is, with proper indexing, the full $G$ will synthesize a consistent value at each pixel location in the image, regardless of the order of the patches are generated. We illustrate the value consistency in Figure 3(c). This characteristic enables the model to independently generate patches that can be combined seamlessly, and maintains a *constant* memory usage while synthesizing images of any resolution.

In practice, aligning each pixel in $\mathbf{z}_{\mathrm{l}}$ to that in the image space can facilitate $\mathbf{z}_{\mathrm{l}}$ and $\mathbf{c}$ indexing. We achieve the goal by shrinking the StyleGAN2 blur kernel size from 4 to 3, causing the model to generate odd-sized features in all layers, due to the convolutional transpose layers.

## E. Model Training

The discriminator $D$ of the InfinityGAN model is similar to the one in the StyleGAN2 method. The detailed architectures of $G$ and $D$ are presented in the supplementary material. The whole network is mainly trained with loss functions from StyleGAN2, the non-saturating logistic loss $\mathcal{L}_{\mathrm{adv}}$ [7], $R_1$ regularization $\mathcal{L}_{\mathrm{R}_1}$ [23] and path length regularization $\mathcal{L}_{\mathrm{path}}$ [13]. Furthermore, to encourage the generator to follow the conditional distribution in the vertical direction, we train $G$ and $D$ with an auxiliary task [26] predicting the vertical position of the patch:

$$\mathcal{L}_{\mathrm{ar}} = \|\hat{\mathbf{c}}_y - \bar{\mathbf{c}}_y\|_1 \,, \tag{5}$$

where $\hat{\mathbf{c}}_y$ is the vertical coordinate predicted by $D$, and $\bar{\mathbf{c}}_y$ is either (for generated images) $\mathbf{c}_y = \tanh(i_y)$ or (for real images) the vertical position of the patch center in the full image. We formulate $\mathcal{L}_{\mathrm{ar}}$ as a regression task.The overall loss function for the InfinityGAN is:

$$\begin{aligned} &\min_{D} \ \mathcal{L}_{\mathrm{adv}} + \lambda_{\mathrm{ar}}\mathcal{L}_{\mathrm{ar}} + \lambda_{\mathrm{R}_1}\mathcal{L}_{\mathrm{R}_1} \,, \\ &\min_{G} \ -\mathcal{L}_{\mathrm{adv}} + \lambda_{\mathrm{ar}}\mathcal{L}_{\mathrm{ar}} + \lambda_{\mathrm{div}}\mathcal{L}_{\mathrm{div}} + \lambda_{\mathrm{path}}\mathcal{L}_{\mathrm{path}} \,, \end{aligned} \tag{6}$$

where $\lambda$'s are the weights.

## 4. Experimental Results

**Datasets.** We evaluate the InfinityGAN method on the Flickr-Landscape dataset consists of 50,000 high-quality landscape images, which are crawled from the Landscape group on Flickr. For the image outpainting experiments, we evaluate with other baseline methods on scenery-related subsets from the Place365 [43] (62,500 images) and Flickr-Scenery [4] (54,710 images) datasets. Note that the Flickr-Scenery here is different from our Flickr-Landscape. For all datasets, we split the data into 80%, 10%, 10% for training, validation, and test. We present the quantitative and qualitative results on the test set only.

**Hyperparameters.** We use $\lambda_{\mathrm{ar}} = 1$, $\lambda_{\mathrm{div}} = 1$, $\lambda_{\mathrm{R}_1} = 10$, and $\lambda_{\mathrm{path}} = 2$ for all datasets. All models are trained with 101×101 patches cropped from 197×197 real images. Since our InfinityGAN synthesizes odd-sized images, we choose 101 that maintains a sufficient resolution that humans can still recognize its content. On the other hand, 197 is the next output resolution if stacking another upsampling layer to InfinityGAN, which also provides 101×101 patches a sufficient field-of-view. We adopt the Adam [14] optimizer with $\beta_1 = 0$, $\beta_2 = 0.99$ and a batch size 16 for

Table 1: **Quantitative evaluation.** Despite we use a disadvantageous setting for our InfinityGAN (discussed in Section A), it still outperforms all baselines after extending the resolution to 4× larger. Furthermore, the user study shows an over 90% preference favors our InfinityGAN results. †The images are first resized to 128 before resizing to 197.

| Method | Resolution | | | FID | ScaleInv FID | | | | Preference v.s. ours | | Inference Memory Complexity |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Full | $G$ Train | 8× Test | $G$ Train | 1× | 2× | 4× | 8× | 4× | 8× | |
| SinGAN [32] | 128 | 128 | 1024 | 22.39 | 22.39 | 73.23 | 150.80 | 214.32 | 0.80% | 1.60% | $\mathcal{O}(\text{resolution}^2)$ |
| COCO-GAN [17] | 128 | 32 | 1024 | 17.52 | 41.32 | 258.51 | 376.69 | 387.15 | 0% | 0% | $\mathcal{O}(1)$ |
| StyleGAN2[13]+NCI | 128 | 128 | 1024 | 5.70 | **5.70** | **24.41** | 83.24 | 201.73 | 9.20% | 7.20% | $\mathcal{O}(\text{resolution}^2)$ |
| StyleGAN2[13]+NCI+FCG | 197† | 101 | 1576 | 86.76 | 90.79 | 126.88 | 211.22 | 272.80 | 0.40% | 1.20% | $\mathcal{O}(1)$ |
| InfinityGAN (Ours) (StyleGAN2[13]+NCI+FCG+$G_S$) | 197† | 101 | 1576 | 13.60 | 27.52 | 35.38 | **62.19** | **115.31** | - | - | $\mathcal{O}(1)$ |

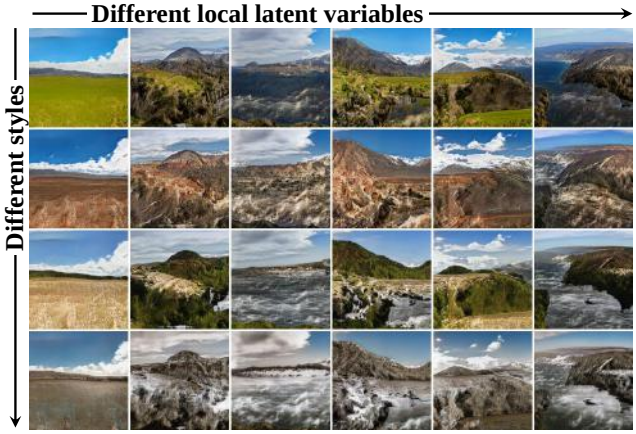


Figure 5: **Generation diversity.** We show that the structure synthesizer and texture synthesizer separately models structure and texture by changing either the local latent $\mathbf{z}_l$ or textural latent $\mathbf{z}_T$ while all other variables are fixed. The results also show that InfinityGAN can synthesize a diverse set of landscape structures at the same coordinate.

800,000 iterations. More details are presented in the supplementary material.

**Metrics.** We first evaluate Fréchet Inception Distance (**FID**) [9] at $G$ training resolution. Then, without access to real images at higher resolutions, we make an assumption that the real landscape with a larger FoV will share a certain level of self-similarity with its smaller FoV parts. We accordingly propose a **ScaleInv FID**, which resizes higher-resolution images to the training data resolution with bilinear interpolation, then computes FID. We denote N× ScaleInv FID when the metric is evaluated with images N× larger than the training samples.

**Evaluated Method.** We perform the evaluation on Flickr-Landscape with the following algorithms:

– **SinGAN.** We train an individual SinGAN model for each image, resulting in 50,000 SinGAN models. The images at higher resolutions are generated by setting spatially enlarged input latent variables. Note that we do not compare with the super-resolution setting from SinGAN since it heavily borrows the structural information from the training image with memorization.

– **COCO-GAN.** Follow the "Beyond-Boundary Generation" protocol of COCO-GAN, we transfer the learned representations from a trained standard COCO-GAN to extended coordinates with a post-training procedure.

– **StyleGAN2 (+NCI).** We replace the constant input of the original StyleGAN2 with a $\mathbf{z}_l$ of the same shape, we call such a replacement as "non-constant input (NCI)". This modification enables StyleGAN2 to generate images at different output resolutions with different $\mathbf{z}_l$ sizes.

## A. Generation at Extended Resolution.

**Additional (unfair) protocols for fairness.** We adopt two additional pre- and post-processing to ensure that InfinityGAN does not take advantage of its different training resolution. To ensure InfinityGAN is trained with the same amount of information as other methods, images are first bilinear interpolated into 128×128 before resized into 197×197. Next, for all testing resolutions in Table 4, InfinityGAN generates at 1.54× (=197/128) higher resolution to ensure final output images share the same FoV with others. In fact, these corrections make the setting disadvantageous for InfinityGAN, as it is trained with patches of 50% FoV, generates 54% larger images for all settings, and requires to composite multiple patches for its 1× ScaleInv FID.

**Quantitative analysis.** For all the FID metrics in Table 1, unfortunately, the numbers are not directly comparable, since InfinityGAN is trained with patches with smaller FoV and at a different resolution. Nevertheless, the trend in ScaleInv FID is informative. It reflects the fact that the global structures generated from the baselines drift far away from the real landscape as the testing FoV enlarges. Meanwhile, InfinityGAN maintains a more steady slope, and surpasses the strongest baseline after 4× ScaleInv FID. Showing that InfinityGAN indeed performs favorably better than all baselines as the testing resolution increases.

**Qualitative results.** In Figure 4, we show that all baselines fail to create reasonable global structures with spatially expanded input latent variables. SinGAN mainly generates a large amount of textural content in the central area while synthesizing at extended resolutions. Similar to StyleGAN2, SinGAN memorizes the structural information with position encoding from zero padding, and tends to

Figure 6: **Spatial style fusion.** We present a mechanism in fusing multiple styles together to increase the interestingness and interactiveness of the generation results. The 512×4096 image fuses four styles across 258 independently generated patches.

Table 2: **Outpainting performance.** The combination of In&Out [4] and InfinityGAN achieves state-of-the-art FID (lower better) performance on image outpainting task.

| Method | Place365 | Flickr-Scenery |
|---|---|---|
| Boundless [34] | 35.02 | 61.98 |
| NS-outpaint [40] | 50.68 | 61.16 |
| In&Out [4] | 23.57 | 30.34 |
| In&Out+ InfinityGAN | **11.61** | **16.87** |

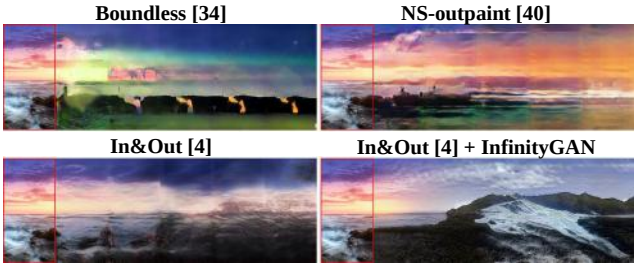


Figure 7: **Outpainting long-range area.** InfinityGAN synthesizes continuous and more plausible outpainting results for arbitrarily large outpainting areas. The real image annotated with red box is 256×128 pixels.

place all memorized objects near the border of the images. COCO-GAN fails to transfer to new coordinates when the extrapolated coordinates are too far away from the training distribution. As for StyleGAN2, the extended input $\mathbf{z}_{\mathrm{l}}$ for generation at the extended resolutions introduces different padding patterns from training, causing StyleGAN2 fails to generate reasonable global structures but generates texture-like patterns. Such a result causes its ScaleInv FID to rapidly surge as the extended generation resolution increases to 1024×1024. Note that at the 16× setting, StyleGAN2 runs out of memory with a batch size of 1 and fails to generate any result. In comparison, InfinityGAN achieves reasonable global structures with fine details. Note that the 1024×1024 image from InfinityGAN is created by compositing 121 independently synthesized patches. With the ability in generating consistent pixel values (Section D), the composition is guaranteed to be seamless. We provide more comparisons, a larger set of generated samples and a very-long synthesis result in the supplementary material.

In Figure 5 and supplementary materials, we switch different $\mathbf{z}_{\mathrm{l}}$ and $G_{\mathrm{T}}$ styles (i.e., $\mathbf{z}_{\mathrm{g}}$ projected with the mapping layer) while sharing the same $\mathbf{c}$. The results show that the structure and texture are cleanly and separately modeled by $G_{\mathrm{S}}$ and $G_{\mathrm{T}}$. The figure also shows that $G_{\mathrm{S}}$ can generate a diverse set of structures by reflecting different $\mathbf{z}_{\mathrm{l}}$.

**User study.** We ask users to select images with more favorable global structures from two sets of 16 images generated from two competing methods. The evaluation is conducted by pairing our InfinityGAN against each of the other baselines. We ask each participant to compare 30 pairs and collect the results from 50 subjects. As shown in Table 1, the user study shows an over 90% of preference favorable to InfinityGAN against all other baselines.

## B. Ablation Studies

As discussed in Section C, we hypothesize that StyleGAN2 memorizes the structure of generated contents with position encoding. In Table 1 and Figure 4, we perform an ablation by removing all paddings from StyleGAN2+NCI, yielding StyleGAN2+NCI+FCG that has no positional information in the generator. The results show that StyleGAN2+NCI+FCG fails to generate reasonable image structures, and significantly degrades in all FID settings. Then, with the proposed $G_{\mathrm{S}}$, the structural information is properly provided from $G_{\mathrm{S}}$, and resumes the generator performance back to a reasonable state.

## C. Applications

**Spatial style fusion.** Given a single global latent variable $\mathbf{z}_{\mathrm{g}}$, the corresponding infinite-resolution image is tied to a single modal of global structures and styles. To achieve greater image diversity and allow the user to interactively generate images, we propose a spatial fusion mechanism that can spatially combine two global latent variables with a smooth transition between them. First, we manually define multiple style centers in the pixel space and then construct an initial fusion map by assigning pixels to the nearest style center. The fusion map consists of one-hot vectors for each pixel, forming a style assignment map. According to the style assignment map, we then propagate the styles in all intermediate layers. Please refer to supplementary material for details. Finally, with the fusion maps annotated for every layer, we can apply the appropriate $\mathbf{z}_{\mathrm{g}}$ from each style center to each pixel using feature modulation.Note that the whole procedure has a similar inference speed as the normal synthesis. Figure 6 shows synthesized fusion samples.

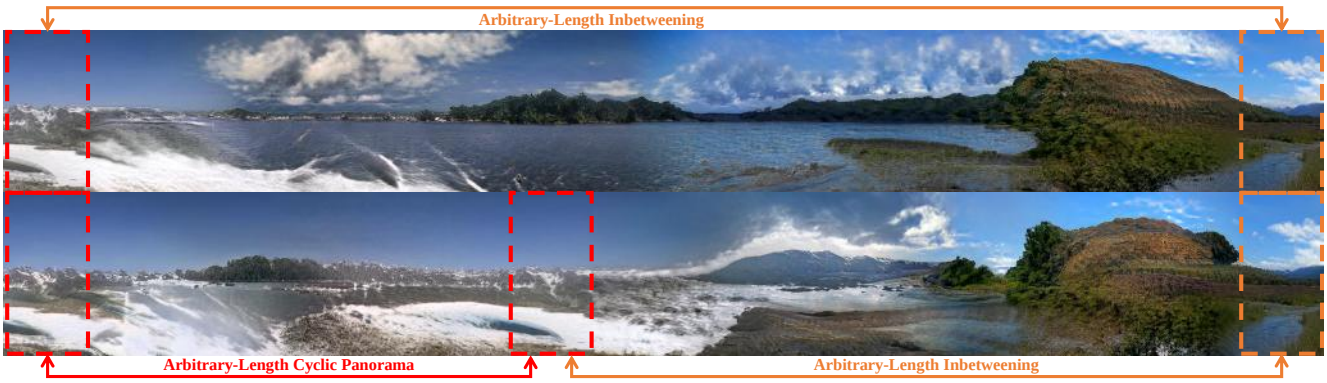


Figure 8: **Image inbetweening with inverted latents.** The InfinityGAN can synthesize arbitrary-length cyclic panorama and inbetweened images by inverting a real image at different position. The top-row image size is 256×2080 pixels.

Figure 9: **Multi-modal outpainting.** InfinityGAN can natively achieve multi-modal outpainting by sampling different local latents in the outpainted region. The real image annotated with red box is 256×128 pixels.

Table 3: **Inference speed up with parallel batching.** Benefit from the spatial independent generation nature, InfinityGAN achieves up to 7.20× inference speed up by with parallel batching at 8192×8192 resolution. The complete table can be found in supplementary materials.

| Method | Parallel Batch Size | # GPUs | Inference Time (second / image) | Speed Up |
|---|---|---|---|---|
| StyleGAN2 [13] | N/A | 1 | OOM | - |
| InfinityGAN (Ours) | 1 | 1 | 137.44 | ×1.00 |
| | 128 | 8 | 19.09 | ×7.20 |

**Outpainting via GAN Inversion.** We leverage the pipeline proposed in In&Out [4] to perform image outpainting with latent variable inversion. All loss functions follow the ones proposed in [4]. We first obtain inverted latent variables $\mathbf{z}'_{\mathrm{g}}$, $\mathbf{z}'_{\mathrm{l}}$ and $\mathbf{z}'_{\mathrm{T}}$ that generates an image similar to the given real image, then outpaint the image by expanding $\mathbf{z}'_{\mathrm{l}}$ to $\hat{\mathbf{z}}_{\mathrm{l}}{}'$ with its unit Gaussian prior. Please refer to supplementary materials for details.

In Table 2, our model performs favorably against all baselines in image outpainting (Boundless [34], NS-outpaint [40], and In&Out [4]). Also, as shown in Figure 7, while dealing with a large outpainting area (e.g., , panorama), all previous outpainting methods adopt a sequential process that generates a fixed region at each step. This introduces obvious concatenation seams, and tends to produce repetitive contents and black regions after the multiple steps. In contrast, with InfinityGAN as the image prior in the pipeline of [4], we can directly outpaint arbitrary-size target region from inputs of arbitrary shape. Moreover, in Figure 9, we show that our outpainting pipeline natively supports multi-modal outpainting by sampling different $\hat{\mathbf{z}}_{\mathrm{l}}{}'$.

**Image inbetweening with inverted latent variables.** We show another adaptation of outpainting with model inversion by setting two sets of inverted latent variables at two different spatial locations, then perform spatial style fusion between the variables. Please refer to supplementary material for details. As shown in Figure 8, we can naturally inbetween [21] the area between two images with arbitrary distance. A cyclic panorama of arbitrary width can also be naturally generated by setting the same image on two sides.

**Parallel batching.** The nature of spatial-independent generation enables parallel inference on a single image. As shown in Table 3, by stacking a batch of patches together, InfinityGAN can significantly speed up inference at testing up to 7.20 times. Note that this speed-up is critical for high-resolution image synthesis with a large number of FLOPs.

## 5. Conclusions

In this work, we propose and tackle the problem in synthesizing infinite-resolution images, and demonstrate several applications of the proposed InfinityGAN framework.

There are still several challenges are left for future study. First is that our Flickr-Landscape dataset consists of images taken at different FoVs and distances to the scenery. InfinityGAN tries to compose landscapes of different scales together, and sometimes leads a bizarre global view. Second is a dataset bias that photographers sometimes include tree leaves on the top of photography. Such a bias sometimes misleads InfinityGAN to synthesize floating islands in the sky. The third is a slight degradation in FID score in comparison to StyleGAN2. This may potentially relate to the convergence problem in video synthesis [35], in which the generator achieves an inferior performance if the motion module is jointly trained with the image module.

# [Supplementary Materials]
# InfinityGAN: Towards Infinite-Resolution Image Synthesis

## Table of Contents

## A. Conceptual Comparisons Among Different Tasks

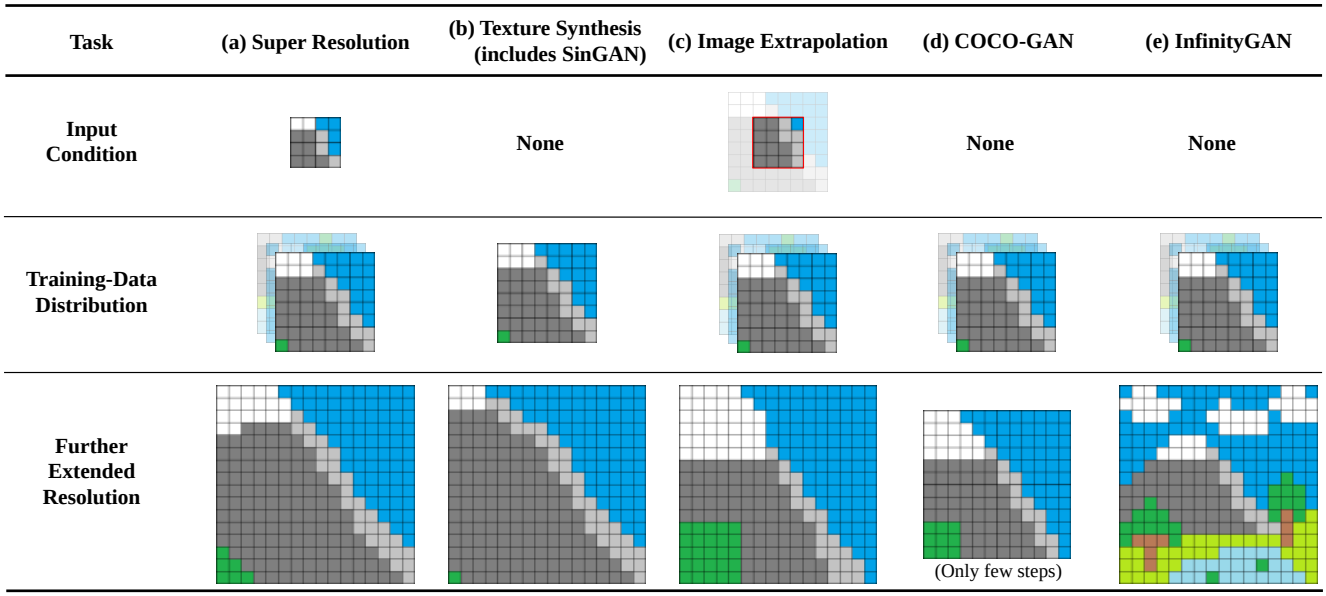


Figure 10: (a) **Super Resolution**: The final outputs inherit the structure from and share the same field-of-view with the original input condition. (b) **Texture Synthesis**: Due to coordinate encoding, objects are generated near image the border, and the center of the image is filled with repetitive textures. (c) **Image Extrapolation**: Current extrapolation models tend to repeat contents on the image border. (d) **COCO-GAN**: COCO-GAN can only synthesize samples slightly larger than its training distribution. (e) **InfinityGAN**: Ours InfinityGAN can synthesize a more favorable global structure at arbitrary resolutions without an input condition.

## B. Implementation Details of Coordinates

We first derive the receptive field size $R$ of an $L$-layer $G_{\mathrm{S}}$ after adding the $7\times7$ feature unfolding to all layers. Assume that the size of $\mathbf{z}_{\mathrm{S}}$ (i.e., output of $G_{\mathrm{S}}$, and input of $G_{\mathrm{T}}$) is $M$. The value of $R$ can be derived as $M + (2\times3)\times L$, where 3 is the half-size of the feature unfolding area. In practice, with an architecture shown in Figure 11 with training patch size $101\times101$, we have $M = 11$, $L = 4$, and $R = 35$.

**Horizontal direction.** In order to avoid the generator discovering and exploiting any property of the periodic coordinate, we use a period $T$ much larger than $R$. Meanwhile, the period should be short enough to avoid the differences between consecutive coordinates vanish to almost zero. In practice, we use $T = 4\times R$ for both of the cosine and sine coordinates.

**Vertical direction.** We exploit the property of the tanh function that its slope rapidly saturates to nearly zero and its value range is bounded by $[-1, 1]$. Such a property is aligned to the property of real-world scenery—the structural interactions among landscape objects are mostly concentrated around the horizon and rapidly saturates to a single modal (i.e., sky, ground, or water) in the vertical direction.

There are two hyperparameters while constructing the tanh coordinates: (i) a pair of cutoff points, and (ii) the sampling period of $\mathbf{z}_{\mathrm{l}}$ in the spatial dimension.

- **Cutoff points** ($c_{\mathrm{cut}}$) are a pair of values that, during training, we only sample the coordinates between the value pair. Such a hyperparameter is required since we are unable to sample all coordinates from an infinitely large coordinate system within the finite training steps. In practice, we set the cutoff points at $\pm0.995$ of tanh-projected coordinate. Note that the underlying effect of using different cutoff point values is not well-investigated, we do not observe obvious changes with slightly different values.
- **Sampling period** ($d_{\mathrm{l}}$) defines the distance between two spatially consecutive values of $\mathbf{z}_{\mathrm{l}}$ are sampled within a training sample. Such a value relates to the grain of the representation that the generator overall models. It is equivalent to the occupation ratio of $\mathbf{z}_{\mathrm{l}}$ between the cutoff points. Thus we can alternatively define a hyperparameter $V$ that defines the occupation ratio of $\mathbf{z}_{\mathrm{l}}$ by $\frac{R}{R+V}$. Then, the sampling period can be derived with $d_{\mathrm{l}} = 2 \times c_{\mathrm{cut}} \times \frac{1}{R+V}$. In practice, we use $V = 10$, resulting in $d_{\mathrm{l}} = 0.1\bar{3}$.

## C. InfinityGAN Architecture

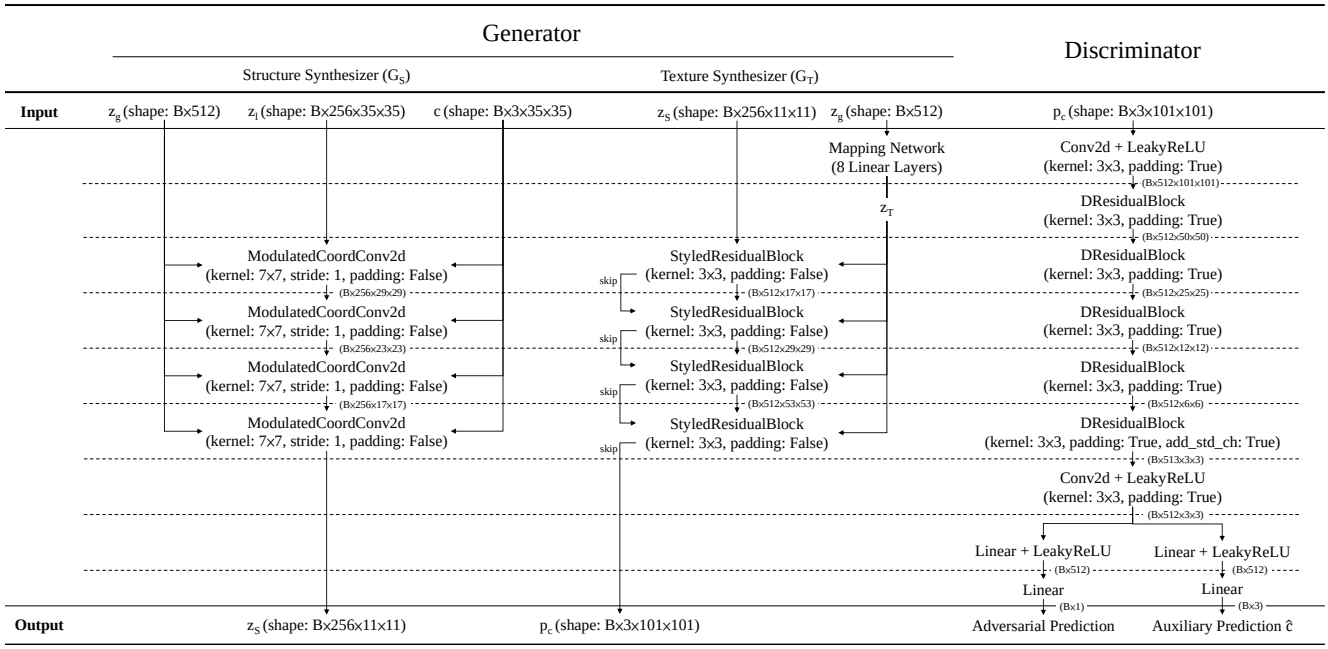


Figure 11: **A high-level overview of InfinityGAN model architecture.**

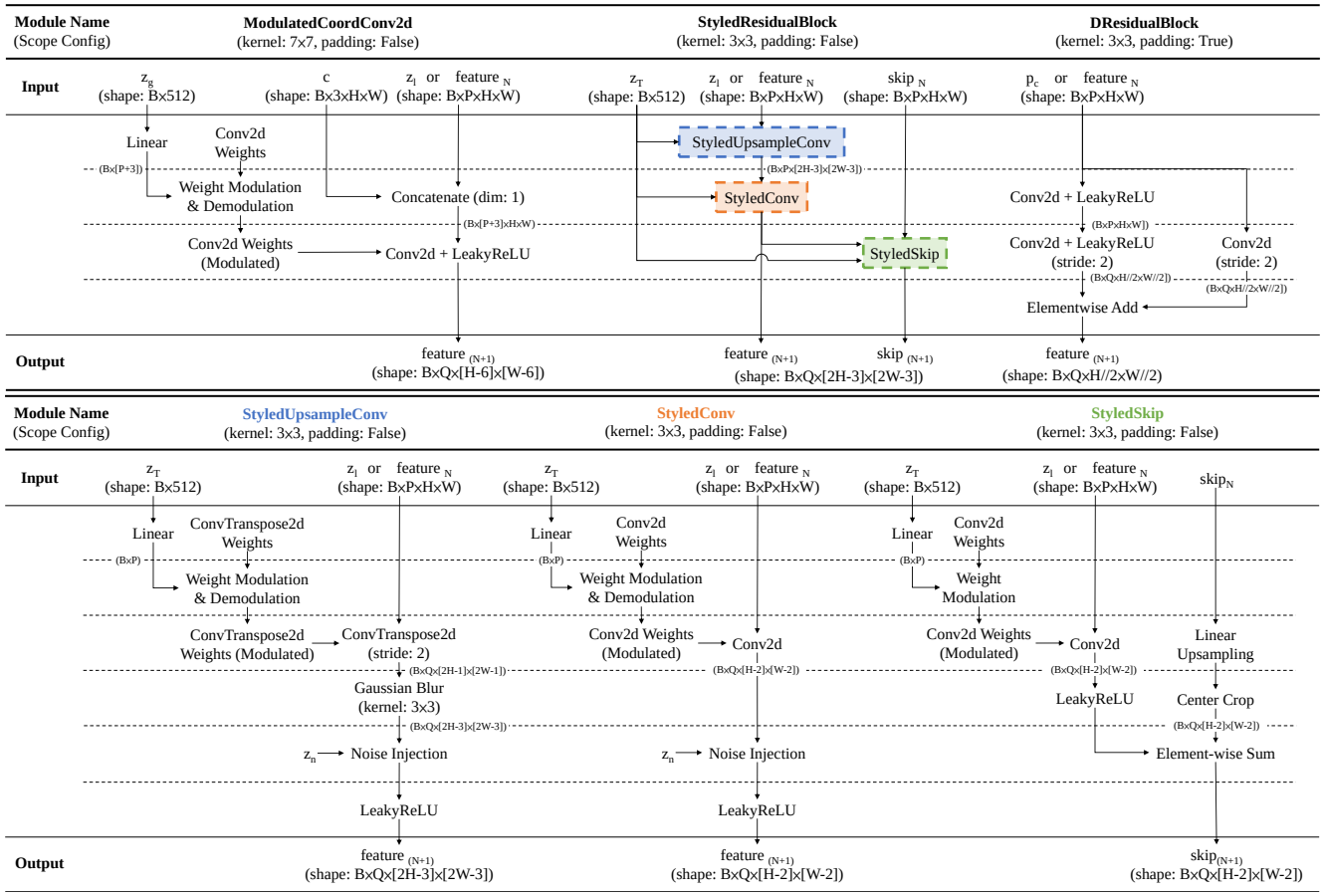


Figure 12: **The low-level design of each module within InfinityGAN.**

## D. More Comparison with baselines

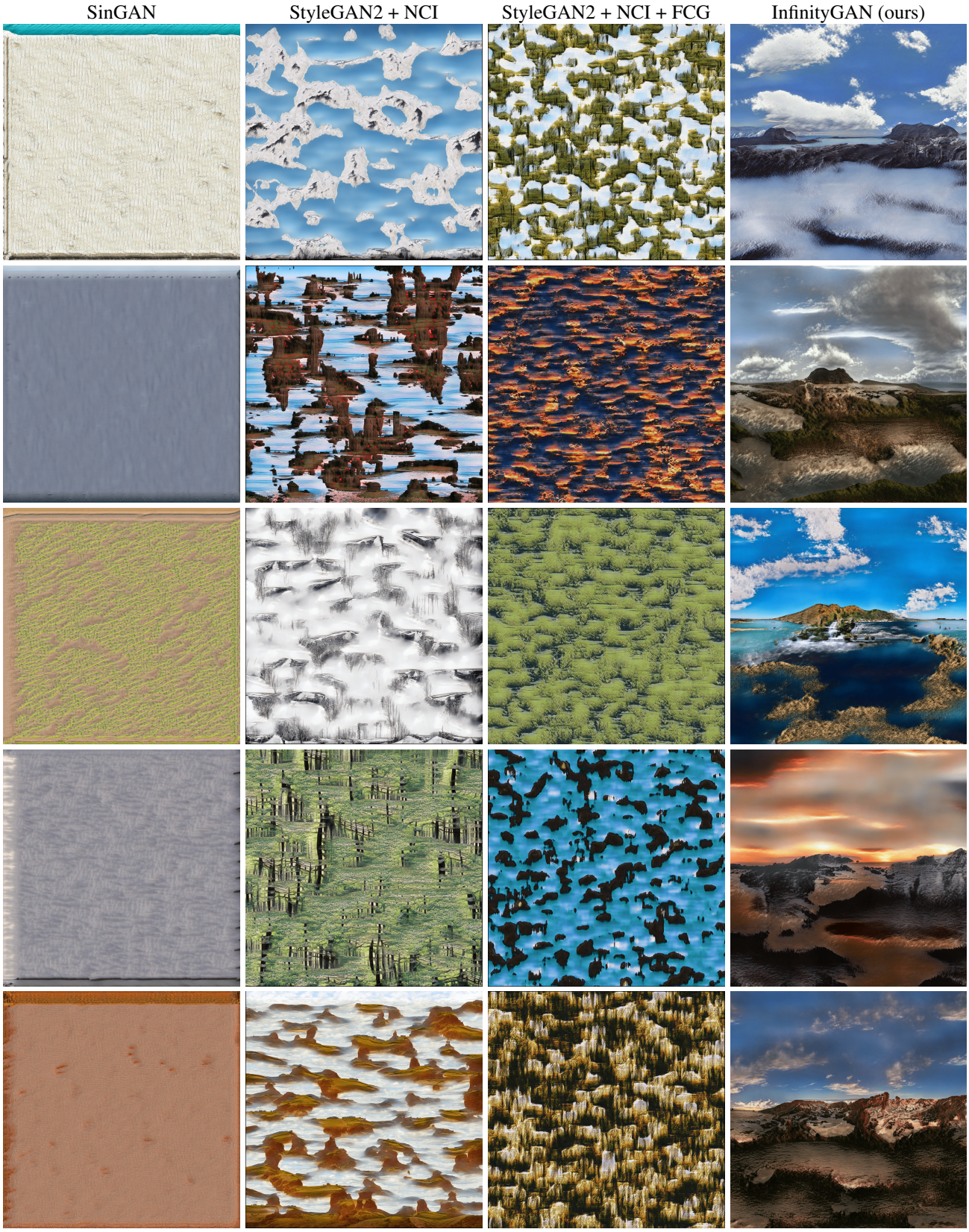


Figure 13: **More qualitative comparisons.** We show more samples on Flickr-Landscape at 1024×1024 resolution.

## E. More InfinityGAN Qualitative Results

Figure 14: **More qualitative results.** We provide more images synthesized at $1024 \times 1024$ resolution with our InfinityGAN trained on Flickr-Landscape. All images are synthesized with the same model presented in the paper, which is trained with $101 \times 101$ resolution patches cropped from $197 \times 197$ resolution real images. All images share the same coordinate and present a high structural diversity.

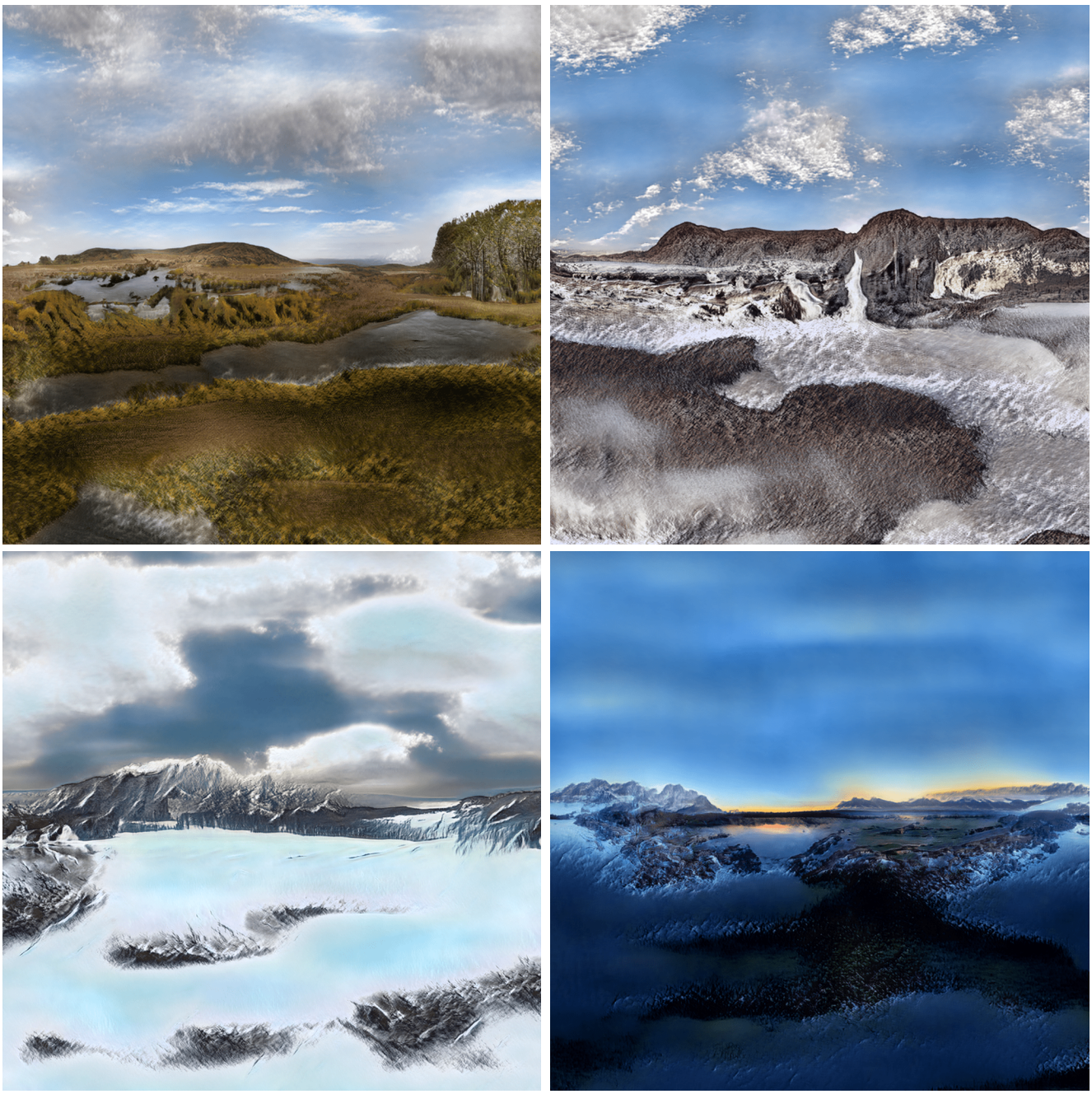

Figure 15: **More qualitative results.** We provide more images synthesized at $1024 \times 1024$ resolution with our InfinityGAN trained on Flickr-Landscape. All images are synthesized with the same model presented in the paper, which is trained with $101 \times 101$ resolution patches cropped from $197 \times 197$ resolution real images. All images share the same coordinate and present a high structural diversity.

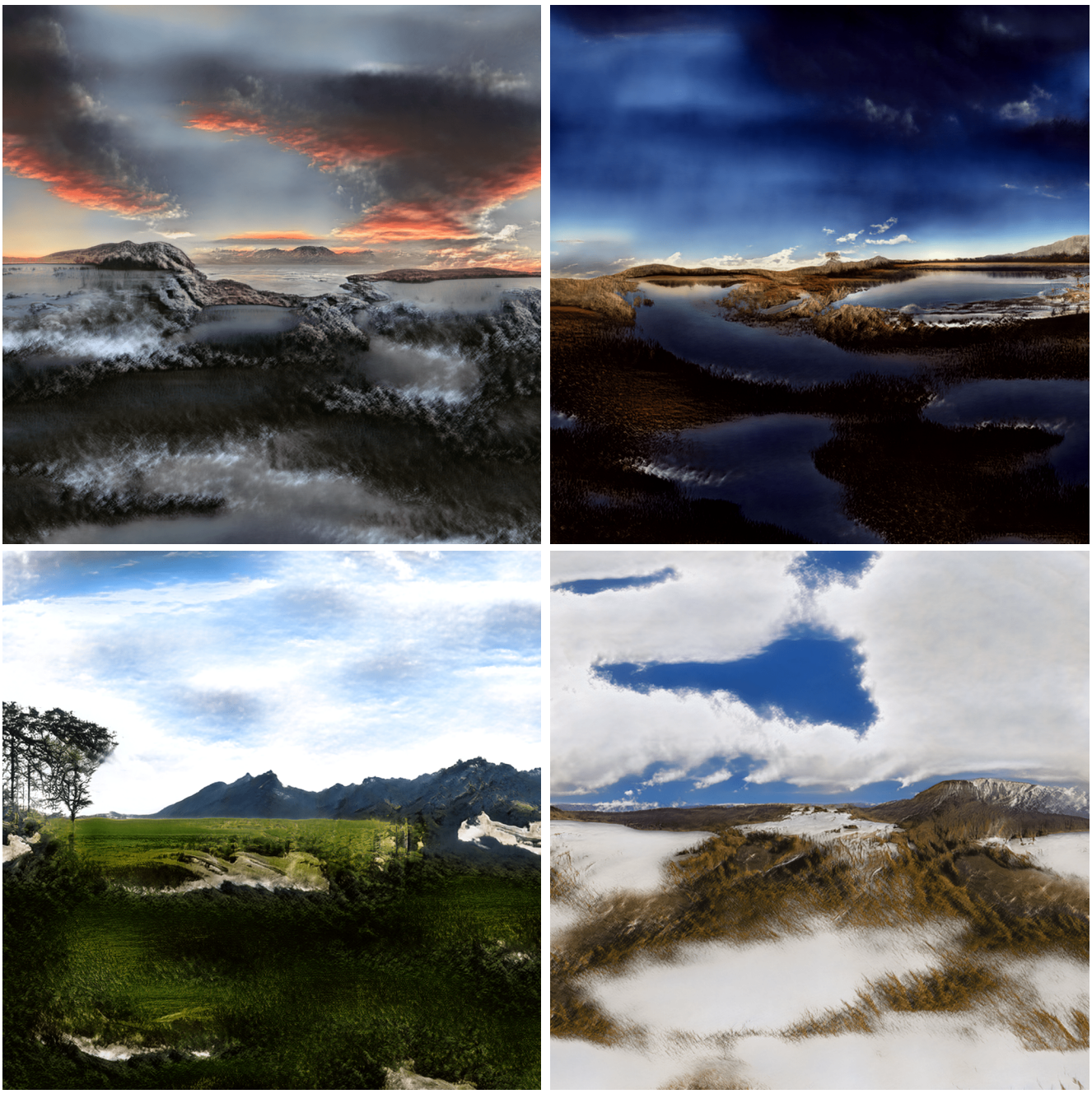

Figure 16: **More qualitative results.** We provide more images synthesized at $1024 \times 1024$ resolution with our InfinityGAN trained on Flickr-Landscape. All images are synthesized with the same model presented in the paper, which is trained with $101 \times 101$ resolution patches cropped from $197 \times 197$ resolution real images. All images share the same coordinate and present a high structural diversity.

## F. More Diversity Visualization

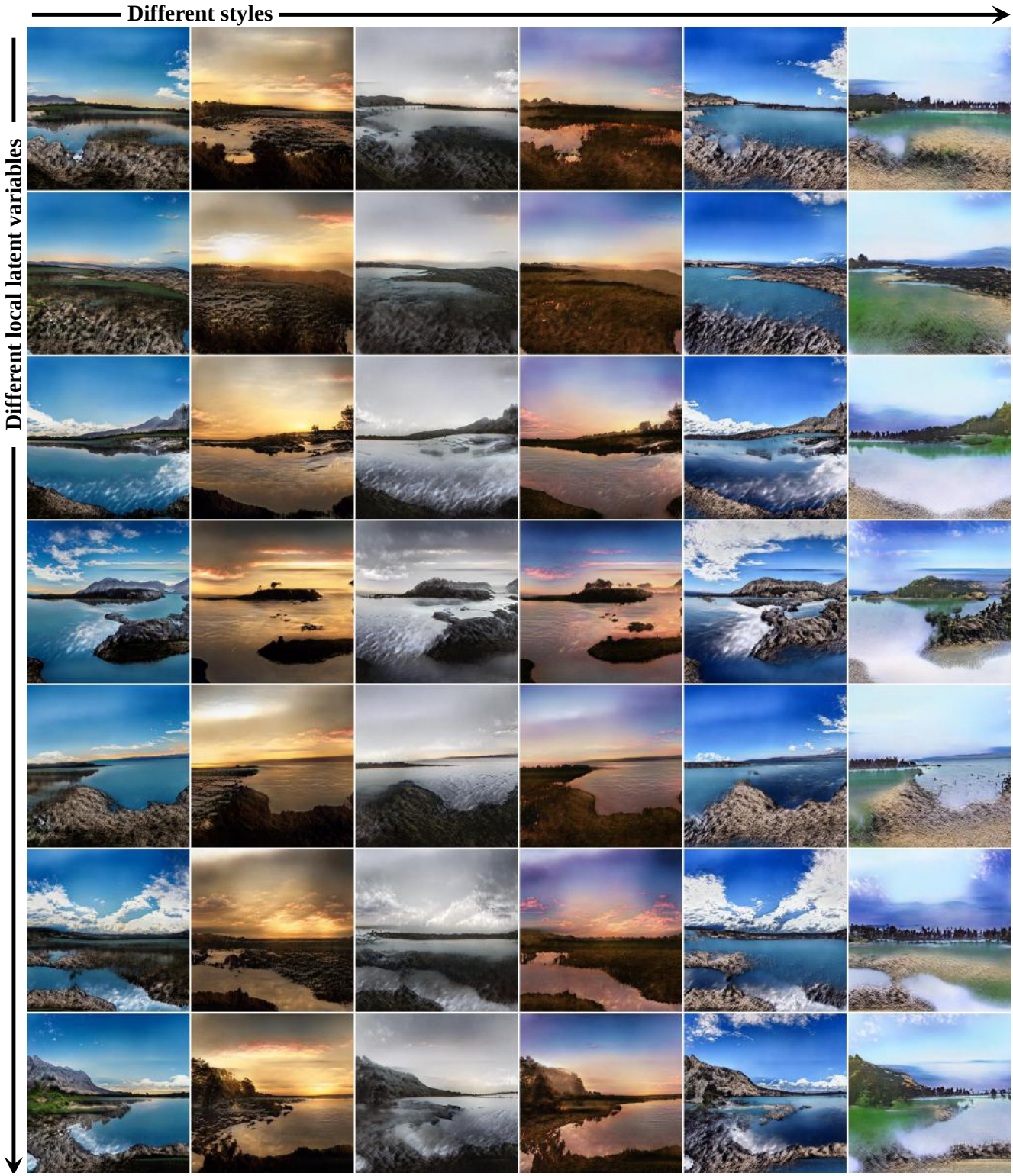


Figure 17: **Generation diversity.** We show that structure synthesizer and texture synthesizer separately models structure and texture by changing either the local latent $\mathbf{z}_{\mathrm{l}}$ or textural latent $\mathbf{z}_{\mathrm{T}}$ while all other variables are fixed. The results also show that InfinityGAN can synthesize a diverse set of landscape structures at the same coordinate. All samples are synthesized at $389\times389$ resolution with InfinityGAN trained at $101\times101$ resolution.

## G. Our Best Attempt in Including A Very-High-Resolution Image


Figure 18: **Very-high-resolution generation.** We provide a 256×9984 resolution sample synthesized with InfinityGAN. The sample shows that (a) our InfinityGAN can generalize to arbitrarily-high resolution, and (b) the synthesized contents do not self-repeat while using the sample global latent variable $\mathbf{z}_{\mathrm{g}}$.

## H. Implementation Details of Spatial Style Fusion

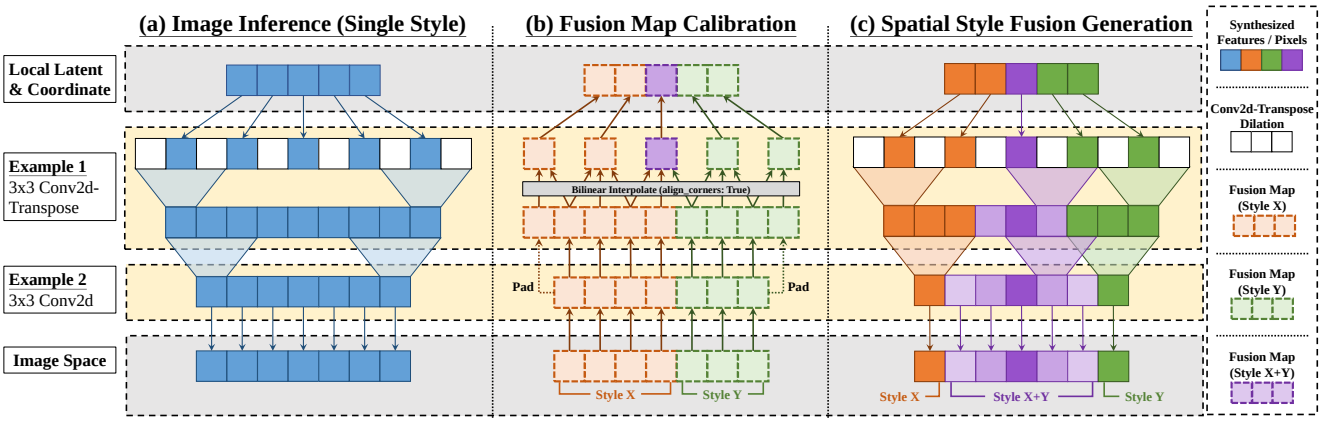


Figure 19: **Illustration of fusion map creation procedure and spatial fusion generation.** With a toy architecture example shown in (a) and a style fusion map in the pixel space (bottom of (b)), we can reversely create spatially aligned fusion maps in all intermediate layers by padding or interpolating the fusion map in the previous layer. Spatial style fusion in (c) uses the fusion maps to synthesize images with a natural style transition in the pixel space.

To achieve spatial style fusion in the existing InfinityGAN pipeline, we introduce two additional procedures: "style fusion map creation" and "fused modulation-demodulation". The former creates per-layer style fusion maps that specify the geometry of the style fusion area. The latter one is a modified version of feature modulation-demodulation that processes the volumetric styles created from the style fusion map.

**Style fusion map creation.** Given $N$ style centers designated by the user, in the pixel space, the target of style fusion map creation is to construct a set of style fusion maps for each layer of both $G_{\mathrm{S}}$ and $G_{\mathrm{T}}$. The fusion map is a spatially-shaped (i.e., batch$\times N\times H\times W$) tensor with $N$ channels that specifies the weight of the style for each spatial location, which the weights sums up to one across the $N$ dimension for each spatial position.

We first construct an initial fusion map in the pixel space by finding the spatially nearest style center, then assign a one-hot label for each spatial position in the initial fusion map. Since the spatial style fusion happens in all layers in the generator, we therefore reversely propagate the fusion map from the output of the generator to its input, we call such a procedure *fusion map calibration*. We show an illustration of fusion map calibration in Figure 19(b). The fusion map calibration starts from the image space and sequentially backward-constructs the fusion maps for all generator layers. For each pair (output-side and input-side) of the fusion maps in a network layer, we match the spatial dimension of the fusion map pair by padding or interpolating the output fusion map into a spatially aligned input fusion map. For different types of intermediate layers, the underlying implementation of the fusion map calibration can be slightly different, but a shared principle is to maintain a consistent geometrical position of the style fusion center throughout the generator.

In practice, such a binary map creates a sharp style transition that produces visible straight lines dividing the style regions. Accordingly, we apply a mean filter that smooths the style transition border. While different kernel sizes for the mean filter only alter the range of style transition and the visual smoothness, we use a kernel size of 127 in our experiments as it empirically produces good visual results.

**Fused modulation-demodulation.** After constructing the per-layer style fusion map, we can use the fusion maps to create volumetric styles (i.e., batch$\times N\times H\times W$) by weighted-sum the styles $\mathbf{z}_{\mathrm{T}}$ by the importance weights ($N$-channel dimension) in each spatial position. The volumetric styles are applied to each layer of the generator. As the feature modulation-demodulation strategy used in both StyleGAN2 and InfinityGAN is a pixel-wise operator, we can easily adapt it to volumetric styles. We demonstrate a possible implementation[1] of the fused modulation-demodulation in Figure 20.

[1]The forward function is based on the implementation from https://github.com/rosinality/stylegan2-pytorch.

```python
import torch
import torch.nn.functional as F

def forward(self, feature, style):
    """
    feature: Feature with shape (B, C1, H, W)
    style  : Single style with shape (B, C2)
    """
    batch, in_c, in_h, in_w = feature.shape


    # Hyperparameters
    k = self.kernel_size # Conv kernel size
    out_c = self.out_c   # Expected output channel
    rmpad = k // 2       # Zero-padding removal

    # Weight scaling (StyleGAN2)
    # Shape:
    # (1, ) * (out_c, in_c, k, k)
    # => (out_c, in_c, k, k)
    weight = self.scale * self.weight

    # Weight modulation (StyleGAN2)
    style = self.modulation(style)
    style = style.view(batch, 1, in_c, 1, 1)
    # Shape:
    # (1, out_c, in_c, k, k) * (batch, 1, in_c, 1, 1)
    # => (batch, out_c, in_c, k, k)
    weight = weight.unsqueeze(0) * style









    # Weight demodulation (StyleGAN2)
    demod = torch.rsqrt(
        weight.pow(2).sum([2,3,4]))
    weight *= demod.view(batch, out_c, 1, 1, 1)









    # Convolution
    feature = feature.view(1, batch*in_h, in_h, in_w)
    if self.upsample:
        weight = weight.view(batch, out_c, in_c, k, k)
        weight = weight.transpose(1, 2).reshape(
            batch*in_c, out_c, k, k)
        out = F.conv_transpose2d(
            feature, weight,
            padding=0, stride=2, groups=batch)

        # Clipping zero padding (ConvT special case)
        out = out[:, :, rmpad:-rmpad, rmpad:-rmpad]









        out = self.blur(out) # StyleGAN2 Gaussian blur
    else:
        weight = weight.view(batch*out_c, in_c, k, k)
        out = F.conv2d(
            feature, weight, padding=0, groups=batch)

    # Recover batch-channel shape due to grouping
    _, _, out_h, out_w = out.shape
    out = out.view(batch, out_c, out_h, out_w)

    return out
```

```python
import torch
import torch.nn.functional as F

def fused_forward(self, feature, style):
    """
    feature: Feature with shape (B, C1, H, W)
    style  : Fusion style with shape (B, C2, H, W)
    """
    batch, in_c, in_h, in_w = feature.shape
    st_c = style.shape[1]

    # Hyperparameters
    k = self.kernel_size # Conv kernel size
    out_c = self.out_c   # Expected output channel
    rmpad = k // 2       # Zero-padding removal

    # Weight scaling (StyleGAN2)
    # Shape:
    # (1, ) * (out_c, in_c, k, k)
    # => (out_c, in_c, k, k)
    weight = self.scale * self.weight

    # Weight modulation (Casted)
    #   The following two forms are equivalent:
    #    - conv(in=feature,         w=weight*style*demod)
    #    - conv(in=feature*style, w=weight) * demod
    #   StyleGAN2 uses the former one for speed.
    style = \
        style.permute(0, 2, 3, 1).reshape(-1, st_c)
    style = self.modulation(style)
    style = style.view(batch, in_h, in_w, in_c)
    style = style.permute(0, 3, 1, 2)
    feature = (style * feature) # (B, C, H, W)

    # Weight demodulation (Approximated)
    #   Feature demodulation use patch statistics.
    #   The approximation here is similar to a
    #   mean of statistics from all styles.
    demod = torch.zeros(batch, out_c, in_h, in_w)
    for i in range(in_h):
        for j in range(in_w):
            style_v = style[:, :, i, j] \
                .view(batch, 1, in_c, 1, 1)
            style_v = weight.unsqueeze(0) * style_v
            # style_v shape: (B, out_ch, in_ch, k, k)
            demod[:, :, i, j] = \
                torch.rsqrt(
                    style_v.pow(2).sum([2,3,4]))

    # Convolution
    # (All feature uses same weight, no need to group)
    if self.upsample:
        weight = weight.view(out_c, in_c, k, k)
        weight = weight.transpose(0, 1).contiguous()
        out = F.conv_transpose2d(
            feature, weight,
            padding=0, stride=2, groups=1)


        # Clipping zero padding (ConvT special case)
        out = out[:, :, rmpad:-rmpad, rmpad:-rmpad]

        # Late demodulation (match output shape)
        demod = F.interpolate(
            demod,
            size=(out.shape[-2], out.shape[-1]),
            mode="bilinear", align_corners=True)
        out = out * demod

        out = self.blur(out) # StyleGAN2 Gaussian blur
    else:
        out = F.conv2d(
            feature, weight, padding=0, groups=1)
        demod = demod[:, :, rmpad:-rmpad, rmpad:-rmpad]
        out = out * demod

    out = out.contiguous()


    return out
```

Figure 20: **Implementation of spatial style fusion.** We present (left) the original StyleGAN2 forward function, and (right) a corresponding implementation for the spatial style fusion. We align the related code blocks on the left and right.

# I. Implementation Details of Image Outpainting and Inbetweening via Inversion

Our pipeline is similar to that of InOut [4]. With a given image $x$, the objective of GAN-model inversion is to recover a set of generator-parameter-dependent input latent variables $z^*$ that can synthesize a resulting image $x^*$ that is similar to $x$. There exist multiple different implementations to recover $z^*$, we adopt the gradient-descent-based method, which optimizes $z^*$ as a set of learnable parameters with carefully designed objective functions. In the context of InfinityGAN, we optimize $\mathbf{z}_\mathrm{g}$, $\mathbf{z}_\mathrm{l}$ for $G_\mathrm{S}$, while $\mathbf{z}_\mathrm{T}$ and $\mathbf{z}_\mathrm{n}$ for $G_\mathrm{T}$. In particular, we uses the $\mathcal{W}^+$ space formulation [1, 38] for $\mathbf{z}_\mathrm{T}$, named $\mathbf{z}_\mathrm{T}^+$, where each layer in $G_\mathrm{T}$ has its own set of ${\mathbf{z}_\mathrm{T}}^i$ and each of the ${\mathbf{z}_\mathrm{T}}^i$ is optimized separately.

**Objective functions.** We first introduce two image-distance losses to the inversion objectives:

$$\begin{aligned} \mathcal{L}_\mathrm{pix} &= \|x - x^*\|_2 \,, \\ \mathcal{L}_\mathrm{percept} &= \mathrm{LPIPS}(x^*, x) \,, \end{aligned} \tag{7}$$

which LPIPS is the learned perceptual distance proposed by Zhang et al._ [42].

We utilize the Gaussianized latent space technique proposed by Wulff and Torralba [38], which uses a LeakyReLU of slope 5 to discount the last activation function (a LeakyReLU with a slope 0.2) in the StyleGAN2 mapping layer, and recovers Gaussian-like marginal distribution of $\mathbf{z}_\mathrm{T}^+$. With the Gaussian distribution prior, we can use the empirical mean $\mu_T^+$ and covariance matrix $\Sigma_T^+$ (computed by sampling 10,000 $\mathbf{z}_\mathrm{T}^+$ via $\mathbf{z}_\mathrm{g}$) to recover an estimated $\mathbf{z}_\mathrm{T}^+$ distribution. With the empirical statistics, [38] proposes to compute Mahalanobis distance:

$$d_M(z, \mu, \Sigma) = (z - \mu)^T \, \Sigma^{-1} \, (\mathbf{z}_\mathrm{T}^+ - \mu). \tag{8}$$

Then, we construct a prior loss $\mathcal{L}_\mathrm{prior}$ [38] that regularizes the $\mathbf{z}_\mathrm{g}$, $\mathbf{z}_\mathrm{l}$ and $\mathbf{z}_\mathrm{T}^+$ with Mahalanobis distance:

$$\mathcal{L}_\mathrm{prior} = \lambda_\alpha d_M(\mathbf{z}_\mathrm{g}, 0, \mathrm{I}) + \lambda_\beta d_M(\mathbf{z}_\mathrm{l}, 0, \mathrm{I}) + \lambda_\gamma d_M(\mathbf{z}_\mathrm{T}^+, \mu_T^+, \Sigma_T^+) \,, \tag{9}$$

which $\lambda_\alpha$, $\lambda_\beta$ and $\lambda_\gamma$ are weight factors. For the cases of $\mathbf{z}_\mathrm{g}$ and $\mathbf{z}_\mathrm{l}$ with zero means and unit variances, the prior loss degenerates to an $l_2$ loss.

Following StyleGAN2, we adopt a noise regularization loss $\mathcal{L}_\mathrm{nreg}$ and noise renormalization. The full objective function of the inversion is:

$$\mathcal{L}_\mathrm{inv} = \lambda_\mathrm{pix}\mathcal{L}_\mathrm{pix} + \lambda_\mathrm{percept}\mathcal{L}_\mathrm{percept} + \mathcal{L}_\mathrm{prior} + \lambda_\mathrm{nreg}\mathcal{L}_\mathrm{nreg} \,, \tag{10}$$

where $\lambda$s are the weighting factor of each loss terms.

**Hyperparameters.** For all tasks and datasets, we set $\lambda_\mathrm{pix} = 10$, $\lambda_\mathrm{percept} = 10$, $\lambda_\mathrm{nreg} = 1000$, $\lambda_\alpha = 10$, $\lambda_\beta = 10$, and $\lambda_\gamma = 0.01$. We use Adam [14] optimizer with a learning annealing [13] from 0.1 to 0 for 3000 iterations. We use a batch size of 1 to avoid batch samples interfere with each other. Despite we observe batched inversion can sometimes yield superior results, it is not a conventional setting for real-world applications that batched inputs are mostly unavailable, it also significantly increases the stochasticity while reproducing the results.

**Outpainting and inbetweening with inverted latent variables.** With the inverted latent variables, we perform image outpainting by spatially extend the $\mathbf{z}_\mathrm{l}^*$ with its unit Gaussian prior, while using $\mathbf{z}_\mathrm{g}^*$ and $\mathbf{z}_\mathrm{T}^{+*}$ everywhere. For image inbetweening, $\mathbf{z}_\mathrm{l}^*$ is also extended with its unit Gaussian prior, while $\mathbf{z}_\mathrm{g}^*$ and $\mathbf{z}_\mathrm{T}^{+*}$ are fused with spatial style fusion. Notice that we do not optimize $c$ in our pipeline, since inverting $c$ is non-trivial and requires additional regularization losses. This introduces a limitation that the spatial position of the images is fixed after the inversion, the users have to perform the inversion optimization again if they want to assign $\mathbf{z}_\mathrm{l}^*$ to a different location.

Another limitation is that, despite the use of the prior loss, some dimensions of the inverted latents tend to drift far away from the normal distribution. In combination with the use of the $\mathcal{W}^+$ space of $\mathbf{z}_\mathrm{T}$, the inverted latents are of high-instability, sometimes introduce checkerboard-like artifacts, and frequently mix multiple irrelevant contexts together. We develop an interactive tool (released along with the code release) that allows users interactively and regionally resampling the undesired local latent variables $\mathbf{z}_\mathrm{l}$ in the outpainting area. Such a limitation is highly related to the generalization of the inverted latents, we put it as an important future working direction.

## J. More Qualitative Results of Outpainting via Inversion

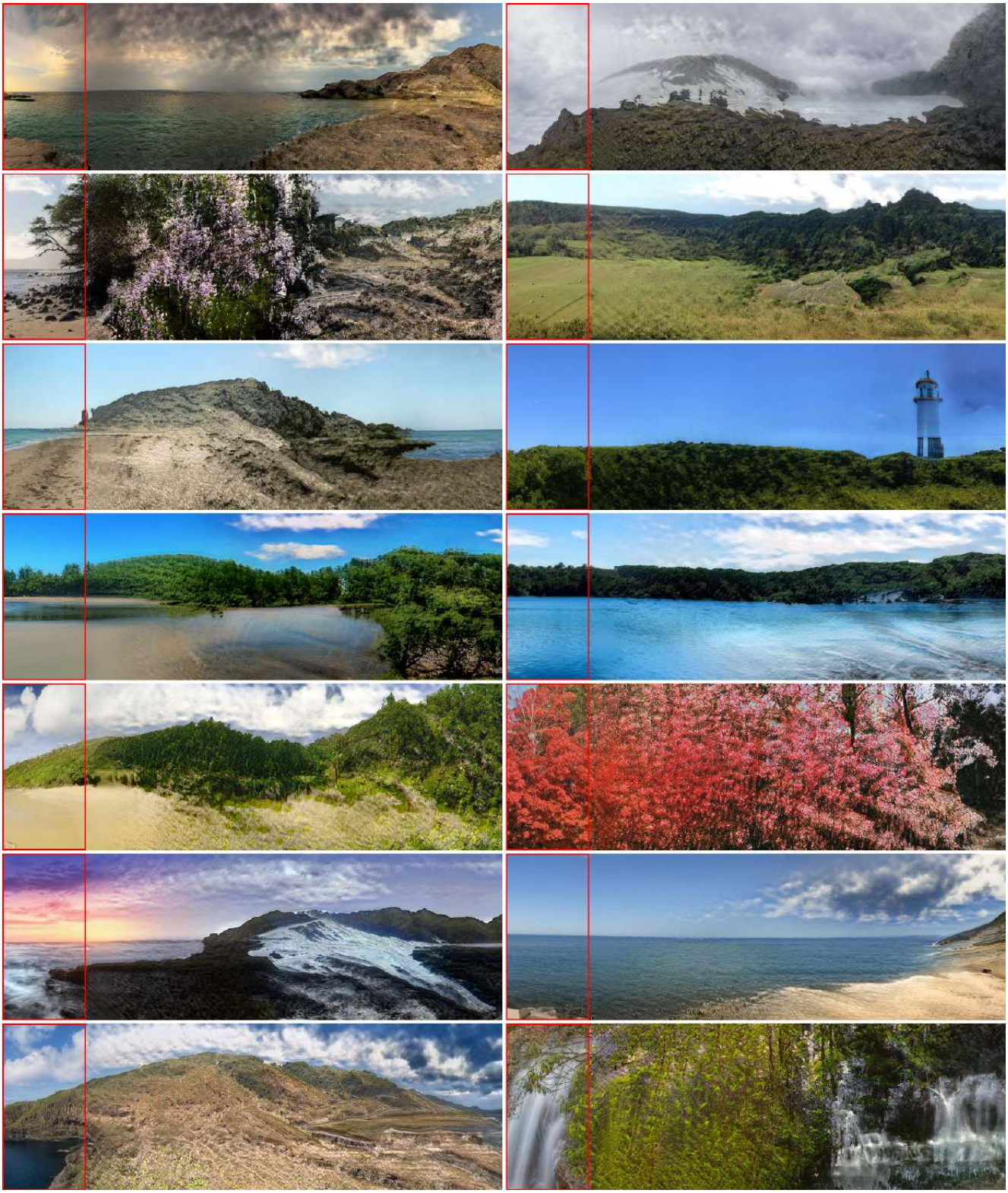

Figure 21: **More outpainting via model inversion.** We present more outpainting results from InfinityGAN on Flickr-Scenery. We invert the latent variables from 256×128 resolution real images (marked with red box), then outpaint 256×640 area (5× real image size).

## K. More Qualitative Results of Inbetweening via Inversion

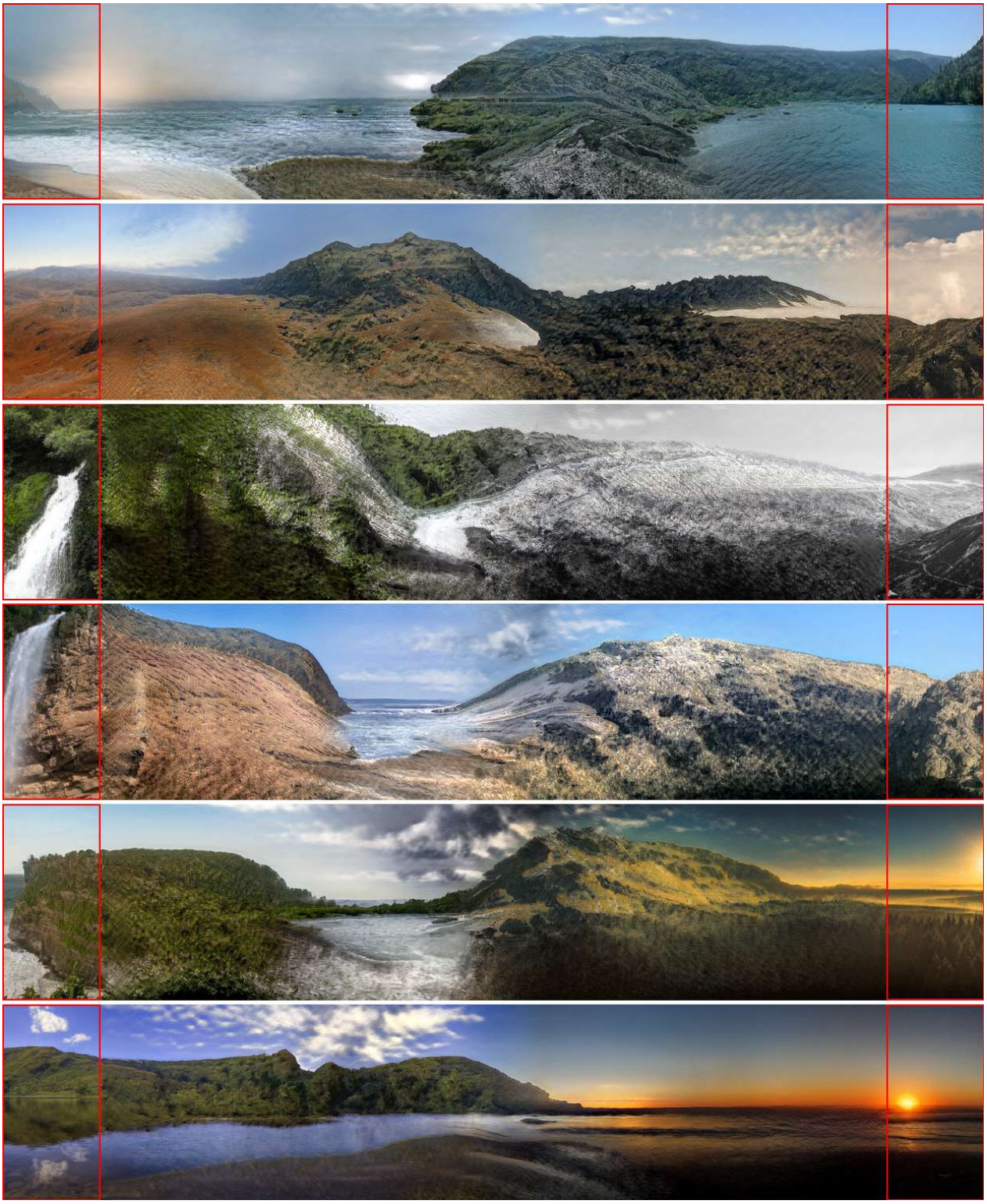

Figure 22: **More image inbetweening with model inversion.** By inverting the latent variables that reconstruct the two real images on two sides (marked with red box), InfinityGAN can naturally inbetween the two images arbitrarily distant away. We synthesize the $256\times1280$ images using InfinityGAN trained on Flickr-Scenery at $101\times101$ resolution.

## L. More Qualitative Results of Cyclic Panoramic Inbetweening via Inversion

Figure 23: **More cyclic panorama synthesized with model inversion.** By setting the same real image on two sides (marked with red box) and inverting the latent variables that reconstruct the real image, InfinityGAN can naturally synthesize horizontally cyclic panoramic images with image inbetweening. We synthesize the $256\times1280$ images using InfinityGAN trained on Flickr-Scenery at $101\times101$ resolution.

## M. Experimental Details of The Speed Benchmark with Parallel Batching

We perform all the experiments on a workstation with Intel Xeon CPU (E5-2650 2.20GHz) and 8 GTX 2080Ti GPUs. We implement our framework with Pytorch 1.6, and execute in an environment with Nvidia driver version 440.44, cuDNN version 4.6.5, and Cuda version 10.2.89.

We report the summation of pure GPU execution time and data scatter-collection time introduced by data parallelism. The model synthesizes a single image for each trial. We first warm up the GPUs with 10 proceeding trials without recording their statistics, then compute the mean and variance over 100 trials. The numbers are reported in Table 4.

Table 4: **Inference speed up with parallel batching.** Benefit from the spatial independent generation nature, InfinityGAN achieves up to 7.20× inference speed up by with parallel batching. We conduct all experiments at a batch size of 1, and OOM indicates out-of-memory. Note that the GPU time here accounts for pure GPU execution time and (if applicable) data-parallel scatter-aggregation time.

| Method | Generation Paradigm | Parallel Batch Size | # GPUs | GPU Time @ Inference Resolution (sec/image) | | | | Speed Up | MFLOPs |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 1024×1024 | 2048×2048 | 4096×4096 | 8192×8192 | 8192×8192 | 1024×1024 |
| StyleGAN2 [13] | One-Shot | - | 1 | $0.60 \pm 0.01$ | OOM | OOM | OOM | - | 6,642 |
| InfinityGAN (Ours) | One-Shot | - | 1 | $0.67 \pm 0.01$ | OOM | OOM | OOM | - | 6,774 |
| | Spatially Independent Generation | 1 | 1 | $1.24 \pm 0.15$ | $7.96 \pm 0.17$ | $34.35 \pm 1.69$ | $137.44 \pm 1.85$ | ×1.00 | 10,799 |
| | | 2 | 1 | $1.58 \pm 0.09$ | $5.31 \pm 0.13$ | $24.13 \pm 0.42$ | $95.77 \pm 1.63$ | ×1.44 | |
| | | 4 | 1 | $1.35 \pm 0.01$ | $5.20 \pm 0.02$ | $20.93 \pm 0.04$ | $82.52 \pm 0.08$ | ×1.67 | |
| | | 8 | 1 | $1.28 \pm 0.01$ | $5.14 \pm 0.02$ | $19.63 \pm 0.02$ | $78.41 \pm 0.17$ | ×1.75 | |
| | | 16 | 1 | $1.23 \pm 0.01$ | $5.01 \pm 0.01$ | $19.11 \pm 0.02$ | $76.41 \pm 0.02$ | ×1.80 | |
| | | 32 | 2 | $0.96 \pm 0.01$ | $3.90 \pm 0.02$ | $14.84 \pm 0.06$ | $59.33 \pm 0.15$ | ×2.32 | |
| | | 64 | 4 | $0.56 \pm 0.01$ | $2.25 \pm 0.05$ | $8.64 \pm 0.11$ | $35.20 \pm 0.39$ | ×3.90 | |
| | | 128 | 8 | $0.32 \pm 0.05$ | $1.30 \pm 0.05$ | $4.82 \pm 0.06$ | $19.09 \pm 0.16$ | ×7.20 | |

## References


[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *CVPR*, 2020. 2, 20

[2] Yogesh Balaji, Mohammadmahdi Sajedi, Neha Mukund Kalibhat, Mucong Ding, Dominik Stöger, Mahdi Soltanolkotabi, and Soheil Feizi. Understanding over-parameterization in generative adversarial networks. In *ICLR*, 2021. 1

[3] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *CVPR*, 2021. 4

[4] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Sergey Tulyakov, Jian Ren, and Ming-Hsuan Yang. Diverse image outpainting via gan inversion. *arXiv preprint arXiv:2104.00675*, 2021. 2, 5, 7, 8, 20

[5] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 2

[6] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arXiv preprint arXiv:2012.09841*, 2020. 2

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 1, 5

[8] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NIPS*, 2017. 1

[9] 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. In *NeurIPS*, 2017. 6

[10] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 3

[11] Md Amirul Islam*, Sen Jia*, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In *ICLR*, 2020. 2, 4

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 1, 2

[13] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 2, 3, 4, 5, 6, 8, 20, 24

[14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5, 20

[15] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2

[16] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2

[17] 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 *ICCV*, 2019. 2, 6

[18] 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. 2

[19] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018. 2

[20] 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. In *NIPS*, 2018. 4

[21] Chia-Ni Lu, Ya-Chu Chang, and Wei-Chen Chiu. Bridging the visual gap: Wide-range image blending. In *CVPR*, 2021. 8

[22] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *CVPR*, 2019. 4

[23] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018. 1, 5

[24] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 3

[25] 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 *ECCV*, 2020. 3

[26] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *ICML*, 2017. 5

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3

[28] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 2

[29] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NIPS*, 2019. 2

[30] Mark Sabini and Gili Rusak. Painting outside the box: Image outpainting with gans. *arXiv preprint arXiv:1808.08483*, 2018. 2

[31] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *CVPR*, 2019. 4

[32] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, 2019. 2, 6

[33] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and retargeting the. In *ICCV*, 2019. 2

[34] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *ICCV*, 2019. 7, 8

[35] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N. Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. In *ICLR*, 2021. 8

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[37] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 2

[38] Jonas Wulff and Antonio Torralba. Improving inversion and generation diversity in stylegan using a gaussianized latent space. *arXiv preprint arXiv:2009.06529*, 2020. 20

[39] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *CVPR*, 2018. 2

[40] Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *ICCV*, 2019. 2, 7, 8

[41] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *ICCV*, 2019. 2

[42] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 20

[43] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017. 5