Efficient Scale-Invariant Generator with Column-Row Entangled Pixel Synthesis

Thuan Hoang Nguyen*   Thanh Van Le*   Anh Tran

VinAI Research, Hanoi, Vietnam

{v.thuannh5,v.thanhlv19,v.anhtt152}@vinai.io

Abstract

Any-scale image synthesis offers an efficient and scalable solution to synthesize photo-realistic images at any scale, even going beyond 2K resolution. However, existing GAN-based solutions depend excessively on convolutions and a hierarchical architecture, which introduce inconsistency and the "texture sticking" issue when scaling the output resolution. From another perspective, INR-based generators are scale-equivariant by design, but their huge memory footprint and slow inference hinder these networks from being adopted in large-scale or real-time systems. In this work, we propose Column-Row Entangled Pixel Synthesis (CREPS), a new generative model that is both efficient and scale-equivariant without using any spatial convolutions or coarse-to-fine design. To save memory footprint and make the system scalable, we employ a novel bi-line representation that decomposes layer-wise feature maps into separate "thick" column and row encodings. Experiments on various datasets, including FFHQ, LSUN-Church, MetFaces, and Flickr-Scenery, confirm CREPS' ability to synthesize scale-consistent and alias-free images at any arbitrary resolution with proper training and inference speed. Code is available at https://github.com/VinAIResearch/CREPS.

1. Introduction

Generative Adversarial Networks (GANs) [8] are one of the most widely used structures for image generation and manipulation [2, 28]. Previously, a GAN model could only generate images with a fixed scale and layout as defined in the training dataset. However, natural images come with varying resolutions and contain unstructured objects at diverse poses. Therefore, designing a generative model that can handle more flexible geometric configurations is gaining more attention in the machine-learning community. StyleGAN3 [13] already supports out-of-the-box translation and rotation with consistent and artifact-free outputs.
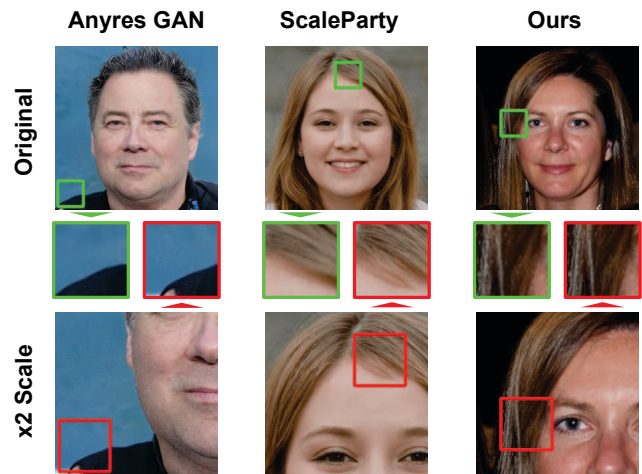

Figure 1. Previous any-scale image synthesis networks, including AnyresGAN [4] and ScaleParty [18], produce inconsistent image details when changing the output scale (see zoomed-in patches). In contrast, our proposed network can produce the same details but sharper when increasing the scale. Check the supplemental video for a clearer comparison.

Any-scale synthesis, however, remains under-explored.

In this paper, we are interested in the task of arbitrary-scale image synthesis where a **single** generator can effortlessly synthesize images at many different scales while **strongly** preserving detail consistency. Such a model can be a promising research direction and bring many benefits. It enables synthesizing a high-resolution image from a lower-resolution training dataset. Hence, it eliminates the need for collecting and training models on high-resolution images, which is costly in storage, time, and computation resources. The output resolution can be ultra-high, e.g., 2048×2048, which is impossible for standard GAN models due to the limit of GPU memory. Any-scale image synthesis also allows geometric interactions like zooming in and out. Despite promising results, previous works on this topic, such as AnyresGAN [4] and ScaleParty [18], show strong inconsistency when scaling the output resolution (see Fig. 1).

We investigate the GAN structures to find the potential cause of the inconsistency at image scaling. Traditional GAN models are based on convolutional generators

*Equal contribution.

[3,15,20], which introduce an implicit spatial bias that helps the model to produce high-quality images. Recently, Xu et al. [29] and Karras et al. [13] discovered that these positional priors can hamper the model's consistency when applying translations, rotations, or scalings. In order to combat this issue, many works introduce non-trivial changes such as sophisticated architecture re-design [13] or opt for a better training strategy and input positional encoding [4,18]. However, these are only partial remedies as the output pixels still depend on their surroundings, making it impossible for these models to produce consistent attributes of an object regardless of positions and scales.

In contrast to the traditional GANs, some recent methods are based on Implicit Neural Representation (INR) [1, 23]. By predicting the color of each pixel separately, INR-based GANs can, in theory, synthesize objects in a spatial-arbitrary manner and still achieve comparable quality at small to medium resolution compared to convolution-based approach. However, these models' memory usage grows quadratically with the input resolution since all pixels have to be queried. Thus, there has been no existing work that can efficiently scale INR-GANs to resolutions higher than 1024. To reduce training complexity, Anokhin et al. [1] employs a simple patch-based strategy where only a portion of pixels is generated and passed through the discriminator at a time. However, this approach unsurprisingly leads to poor results and inconsistency between patches.

Inspired by the latter approach, we aim to tackle the task of scale-consistent image generation with essential changes to StyleGAN2 [15]. Similar to Anokhin et al. [1], we change the 3×3 convolutions to 1×1 ones and add a Fourier feature embedding [25] at the input layer. Although these two changes alone already achieve our goal, it is still expensive to train in high-resolution settings. Thus, instead of using dense 2D features, our model relies on a novel thick bi-line representation, which largely reduces the training and inference complexity by using two low-rank features for row and column. Our network first regresses these row and column embeddings, then composes layer-wise intermediate 2D features, and finally fuses these maps to produce the final output. We name this novel structure **C**olumn-**R**ow **E**ntangled **P**ixel **S**ynthesis, or **CREPS** for short.

We run a series of experiments on four datasets, including FFHQ, MetFaces, LSUN-Church, and Flickr-Scenery, to confirm the effectiveness of our proposed CREPS structure. Our model can synthesize images with quality comparable to the previous generative models like CIPS or StyleGAN2. While CIPS has trouble in training on images of resolutions more than 256×256, CREPS can sufficiently handle training data at resolutions 512×512 and 1024×1024. CREPS produces scale-equivariant images and keeps the object details unchanged when scaling the output resolution, unlike previous any-scale GANs such as AnyresGAN

and ScaleParty. Using a CREPS model trained on 512×512 images, we still can generate near-realistic images at higher resolution. Finally, we demonstrate CREPS's ability to synthesize images with complex geometric transformations and distortions while preserving attribute consistency.

To summarize our contributions:

- We propose a simple and elegant network equipped with only modulated linear layers and no upsampling layers in-between. It supports scale-consistent outputs for any-scale image synthesis.

- To further improve efficiency, we introduce a thick bi-line representation, which decomposes 2D network features into two light-weight row and column embeddings. It significantly saves memory and computation costs compared with the full 2D-feature counterparts.

- We demonstrate competitive results for unconditional image synthesis on the FFHQ, LSUN-Church, Met-Faces, and Flickr-Scenery datasets, along with the ability to generate each image at arbitrary scales with consistent details.

- Our CREPS models support complex geometric transformations and distortions.

2. Related Work

Generative Adversarial Networks. Prior to denoising diffusion models [10, 24], GANs [8] hold state-of-the-art results for image synthesis tasks. The popular GAN models can generate realistic images at a high resolution, commonly up to 1024×1024 [3, 12–15]. The promising results obtained by GANs have motivated several applications of computer graphics and visual content generation. However, these networks are only capable of generating images with same geometric configurations, e.g., center-located and face-forward objects. Recently, an exciting work StyleGAN3 [13] aimed to generalize GAN to arbitrary translation and rotation with consistent details, or Anycost GAN [16] with multi-resolution generation. In the same vein, ScaleParty [18] and AnyresGAN [4] extended Style-GAN2 and StyleGAN3 to support scaling and other geometric transformations by replacing learned input constant with suitable positional encoding and multi-scale training strategy. However, these works did not consider the scale consistency, and their images showed varied details as the output scale increases, illustrated in Fig. 1.

Implicit Neural Representation. Typically, images are represented by a series of 2D arrays of values. However, it can be viewed as a continuous mapping from a 2D coordinate $(x, y) \in \mathbf{R}^2$ to the corresponding RGB value $(r, g, b) \in \mathbf{R}^3$ and the mapping can be parameterized as a

black-box model. This coordinate-wise modeling has been used in a wide range of neural rendering tasks [5,17,22,26], where neural networks are used to provide an efficient and continuous representation of data compared with traditional methods. In the literature, implicit neural networks mainly utilize fully-connected layers as their building blocks. Unlike convolution or self-attention, such layers' receptive field size is exactly one; in other words, the output at every coordinate is independent of each other.

INR-based GANs. As the number of research increased, INR started to be used for generative tasks. These models soon inherited the success of GANs by employing the adversarial training manner. Generative radiance fields [6,9,19,21] attempt to learn a view-consistent representation of 3D objects using implicit GAN. Despite all the success of INRs in 3D GANs, limited attention has been paid to utilizing the equivariance capability of fully-connected layers in 2D counterparts. The closest to our work are INR-GAN [23] and CIPS [1]. Both these works use a grid of the target pixel coordinates as input for batch processing instead of passing each point individually. INR-GAN employs a multi-scale structure, which we will discuss later as a cause of scale inconsistency, while its uniform-scale versions have poor generation outputs. Meanwhile, CIPS does not need the multi-scale design thanks to its efficient weight modulation and expressive input embedding. The uniform-scale INR-GAN and CIPS disregard spatial convolutions in the generator and synthesize each pixel independently. However, their main goal is to investigate an alternative architecture that can compete with fully-convolutional GANs rather than paying attention to the equivariance characteristic of such models. They also struggle with expensive computation costs and memory usage using full-resolution 2D feature maps in processing.

3. Proposed method

This section describes our proposed CREPS structure. First, we recall the concept of any-scale image synthesis (Sec. 3.1). Then, we revise two existing GAN structures that support scale-equivariant image synthesis (Sec. 3.2). Next, we discuss how to reduce computation cost via the novel thick bi-line representation (Sec. 3.3). Finally, we describe the layer-wise feature composition scheme for improving the synthesis quality (Sec. 3.4).

3.1. Any-scale image synthesis

In this section, we introduce any-scale image synthesis as the task of generating images while enforcing consistency at different scales given a single model. One way we naturally come up with is generating an image at many scales altogether. MSG-GAN [11] is one of the earliest works in this approach. Instead of producing single output,

MSG-GAN outputs an RGB image at each block of the generator, resulting in a mipmap representation [27]. However, this approach can only output pre-defined discrete scales, and there is no mechanism to guarantee scale consistency.

As such, we should consider injecting positional encoding e as an additional input alongside the latent code into the generator. This approach is employed in some previous works [1, 4, 18, 23], in which e is a 2D grid of normalized (x, y) coordinates. If e is a regular grid, we can decompose it into two vectors for the row and column coordinates denoted as e^r and e^c, respectively. The image generation process now becomes:

$$I = G(z, e^r, e^c), \tag{1}$$

with G is the generative model and z is the latent input. The decomposition from e to e^r and e^c is more suitable to our thick bi-line representation, as later discussed. Doing so allows us to easily control the output's scale and other spatial properties via appropriate input encoding. However, naively adding positional input into an existing generator does not guarantee that the output image is equivariant to the change in the input coordinates. For example, when Karras et al. [13] replace the learnable constant in Style-GAN2's input layer with Fourier features (Config B), the "texture sticking"' issue still occurs. Therefore, proper network design and training strategy should be examined to alleviate the output's geometric inconsistency.

3.2. Removing coarse-to-fine design and spatial convolution

We investigate two network structures that support any resolution image generation, including AnyresGAN [4] and CIPS [1], when keeping the same latent input but gradually increasing the output resolution. The former is built upon StyleGAN3 [13] with additional scale information concatenated with the latent code and a multi-scale training scheme. In contrast, the latter changes the 3x3 convolution of Style-GAN2 with a point-wise one and adds learnable Fourier features at the beginning. Both models are capable of multi-scale generation, but they have different behaviors that we will discuss below.

As illustrated in Fig. 1, while having good photo-realism, AnyresGAN produces different image details at different scales. This can be explained by the fact that AnyresGAN, similar to most other GAN-based works, relies on spatial convolutions, such as 2D convolution with kernel size 3×3 and upsample layers. When changing the output resolution, the neighbor pixels at each location change, greatly varying the output of this spatial-convolution-based network.

On the other hand, CIPS keeps the output image's details nearly same regardless of resolution, thanks to its spatial-free building operators. CIPS, however, is very computationally expensive; this can be clearly shown in Tab. 1.
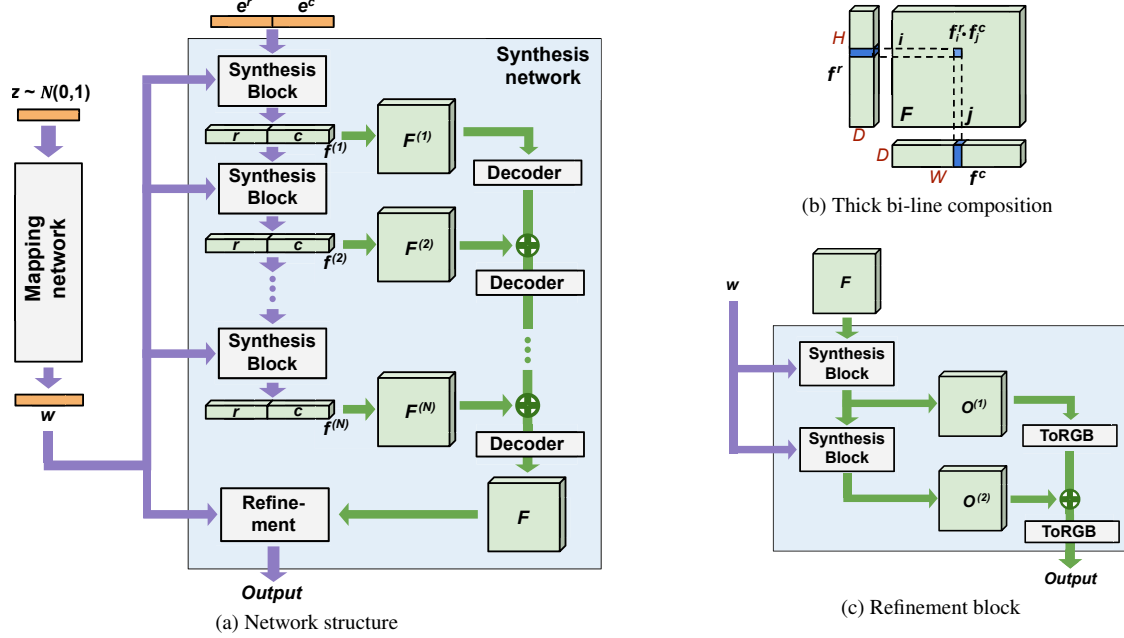
22410

(a) Network structure

(b) Thick bi-line composition

(c) Refinement block

Figure 2. Our proposed CREPS structure

Resolution	Batch size	Memory Usage			Running time		
		StyleGAN2	CIPS	Ours	StyleGAN2	CIPS	Ours
256×256	1	1.5GB	3.3GB	2.3GB	0.04s	0.06s	0.03s
	4	2.5GB	10.2GB	5.2GB	0.05s	0.23s	0.06s
512×512	1	1.7GB	10.4GB	4.5GB	0.04s	0.21s	0.05s
	4	3.4GB	OOM	14.6GB	0.06s	OOM	0.16s

Table 1. Memory usage and running time comparison between StyleGAN2, CIPS and our method. OOM means out-of-memory.

When measured on a single NVIDIA V100 GPU (32 GB) and all models have comparable number of parameters, it runs slower than StyleGAN2 as well as requires much more memory or even gets an out-of-memory (OOM) error when running at 512×512 resolution. This makes CIPS inapplicable to use for learning fine details from high-resolution datasets. Moreover, it is worth noting that our method achieve the best trade-off between speed and memory.

Based on the above observations, we implement CREPS without any spatial convolutions or coarse-to-fine design. Starting with StyleGAN2 [15], which consists of a mapping network and a generator, we remove all upsampling operators and replace all spatial convolutions with 1×1 convolutions, which are equivalent to pixel-wise fully-connected layers. Next, we replace the constant in the first synthesis block with Fourier encodings of the input coordinate row and column e^r and e^c. This design is quite similar to CIPS, with only two minor differences. Firstly, the dense 2D grid input is now split into two vectors representing the row and column. Secondly, we do not combine learned input constant with the Fourier feature like CIPS did, making our

model simpler and more memory-friendly. While this initial network guarantees any-scale image synthesis with consistent image details, it faces the same memory issue as CIPS. We will discuss next how to solve this issue effectively.

3.3. Thick bi-line representation

Inspired by the tri-plane representation in [6], we propose to decompose each feature with 2D spatial dimensions into a column and a row embedding for a memory-efficient representation. For simplicity, let us drop the first two dimensions for the batch size B and the number of channel C, which are the same and element-wise processed for both the feature map and the mentioned embeddings. Let us denote the feature map as $F \in \mathbb{R}^{H \times W}$, with H and W as the height and width, respectively. We can decompose F to a row embedding f^r and a column embedding f^c. In the simplest form, f^r and f^c are 1D vectors with the lengths H and W, respectively. Each pixel in the feature map F_{ij}, with i and j as the row and the column indices, can be computed as the product of the corresponding elements in f^r and f^c:
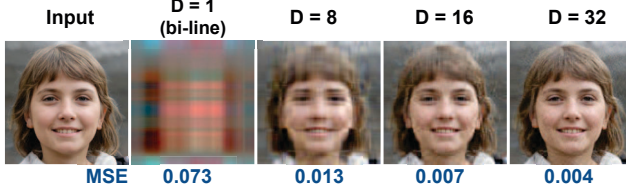
$$F_{ij} = f^r_i f^c_j. \qquad (2)$$

22411

Figure 3. Fitting an input image to the thick bi-line representation in the image space.

We call this representation "bi-line", which significantly reduces the memory usage and computation cost and allows the network to learn with high-resolution data. However, we found that this simple representation had a limited capacity and could not define complex structures. To enrich its representation power, we "thicken" the embeddings by adding an extra, short dimension. The revised f^r and f^c now have the shapes $H \times D$ and $W \times D$, respectively, where $D \ll min(H, W)$ is a uniform embedding "thickness". The composing feature element F_{ij} is now the dot product of the corresponding elements in f^r and f^c:

$$F_{ij} = f_i^r \cdot f_j^c = \sum_{d=1}^{D} f_{id}^r f_{jd}^c. \tag{3}$$

This composition process is illustrated in Fig. 2b. In another perspective, this can be considered as sum of D different bi-line compositions. We call it "thick bi-line" representation.

In Fig. 3, we provide a toy example illustrating the capacity of the proposed thick bi-line representation. Given an input image at resolution $512 \times 512 \times 3$, we fit it into the proposed bi-line representation in the image space by optimizing a row and a column embedding of shape $512 \times D \times 3$. Note that each channel is optimized independently. As can be seen, with the naive bi-linear composition ($D = 1$), the reconstructed image is just a simple, incomprehensible grid. By adding just a small thickness $D = 8$, we can capture the essential image content, recover the subject's identity, and reduce the MSE almost 6 times. When using $D = 32$, we nearly recover the original image with only subtle pixel noise. Note that the row and column embeddings only take 1.56% of the original image size when $D = 8$ and 12.5% when $D = 32$. This experiment confirms the efficacy of our proposed thick bi-line decomposition. Also, while this representation does not capture all details of the complex input image, it is more sufficient when modeling the over-parameterized feature space.

3.4. Layer-wise feature composition

In CREPS, we assume the target output is square, i.e., $H = W$. Hence, we can concatenate the row and column embeddings to a single tensor $f = [f^r, f^c] \in \mathbb{R}^{H \times 2D}$. Initially, we implement CREPS by revising StyleGAN2's code to predict f from the latent input w via N synthesis blocks.

The network then splits f to get the row and column codes, perform the feature composition defined in Eq. (3) to get a feature map F. This feature map will be passed to a simple refinement module (Fig. 2c) with 2 synthesis blocks to produce the output image. For efficient memory and computation cost, we only employ a small thickness value $D = 8$.

We found this initial design needed to be more efficient to catch up with the generation quality of StyleGAN and CIPS. It performed feature composition once near the end of the image synthesis process; thus, the model power was bounded by the capacity of the thick bi-line representation. Instead, we revise our solution by employing a layer-wise feature composition scheme. Specially, at each layer with index $l \in [1..N]$, we extract the intermediate row and column embedding $f^{(l)}$. We can split $f^{(l)}$ and compose an intermediate feature map $F^{(l)}$, following Eq. (3). Then, the intermediate maps across layers are fused to get the final map F. This scheme enriches the representation power, similar to when increasing D while using less memory.

The fusion scheme is also important. Intuitively, we can set F as the sum of the intermediate maps $\{F^{(l)}\}_{l=\overline{1,N}}$. However, this formulation treats the maps equally, and we find it undesirable. Let us call back the StyleGAN models' behavior. Thanks to the coarse-to-fine design, their early layers learn to capture the global shape, while the later layers learn to synthesize fine details. Since CREPS has no coarse-to-fine structure, it is hard to control which aspect of the output image each layer can learn. Hence, we propose adding asymmetry to the feature map fusion process: the feature maps at earlier layers are processed "deeper" than those at later layers. We hope it implicitly guides the layers to learn information from global to regional order, similar to StyleGAN. To do so, we introduce at each layer with index l a narrow decoder, denoted as $\pi^{(l)}$. The process to fuse the intermediate maps $\{F^{(l)}\}_{l=\overline{1,N}}$ is defined as following:

$$E^{(1)} = F^{(1)}, \tag{4}$$
$$E^{(l+1)} = \pi^{(l)}(E^{(l)}) + F^{(l+1)} \quad \forall l \in [1, N-1], \tag{5}$$
$$F = \pi^N(E^N), \tag{6}$$

with $E^{(l)}$ records the fused feature map at the lth layer. In our implementation, each decoder consists of pixel-wise fully-connected layers with Leaky-RELU activations. Fig. 2a illustrates our proposed network structure, while Tab. 1 illustrates the efficiency of our proposed structure in memory usage and running time.

4. Experiments

4.1. Experimental setup

Datasets. We conduct experiments on the common datasets when benchmarking CREPS, including FFHQ, MetFaces,

Figure 4. Sample images with our models trained on FFHQ (upper-left), LSUN-Church (upper-right), MetFace (bottom-left), and Flickr-Scenery (bottom-right).

LSUN-Church, and Flickr-Scenery. FFHQ dataset contains 70k high-quality, diverse human faces collected from Flickr. We will use the FFHQ images with resolution 512×512. MetFaces is a small dataset of face drawings extracted from the collection of the Metropolitan Museum of Art, with a total of 1336 images at resolution 1024×1024. LSUN-Church consists of 126k outdoor photographs of churches at the resolution 256×256. Finally, Flickr-Scenery [7] is a landscape-centric dataset collected on Flickr with 50k images at resolution 256×256.

Implementation. We use StyleGAN2 network design as a reference to implement CREPS. Except for the refinement module, our generator consists of 6 (for the target resolution 256) to 8 synthesis blocks (for the resolution 1024) and the same number of decoder blocks. We replace all modulated convolution layers in StyleGAN2 with modulated fully-connected ones. Unlike StyleGAN2, the output of each block is not an RGB image but a 32-channel bi-line feature with thickness $D = 8$. Each decoder is a stack of $P = 4$ pixel-wise fully-connected layers, with the channel widths ranging from 32 to 128. This setting is applied for all experiments, except for our ablation study. Similar to StyleGAN3 and CIPS, we turn off style mixing regularization. Besides that, we kept most of the other components unchanged, including the mapping network, discriminator, path length regularization, and R_1 gradient penalty.

Training. For FFHQ and LSUN-Church, our networks were trained from scratch until convergence. To verify the flexibility and scale consistency of CREPS on higher-

resolution image synthesis, we increase the length of its coordinate input to generate images at resolution 1024×1024 on the FFHQ dataset. We also test the adaptability of our network on domain shift by applying transfer learning from the weights trained on FFHQ to MetFaces. Our networks were trained by Adam optimizer with learning rate 2×10^{-3} and hyperparameters $\beta_0 = 0$, $\beta_1 = 0.99$, and $\epsilon = 10^{-8}$. We use 4 NVIDIA A100 40GB GPUs for training all models.

4.2. Image generation

Tab. 2 compares the quality of images generated by our CREPS models with the standard spatial-convolution-based StyleGAN2 and the only scale-consistent any-scale image generation technique CIPS, using the Frechet Inception Distance (FID) score.

At resolution 512×512 on FFHQ, our model achieves the FID score of 4.43, which is much better than the score from CIPS (6.18) and not far from StyleGAN2 (3.41). We can also use this model to generate images at resolution 1024×1024 without retraining and achieve a better FID score (4.09). We found that CIPS cannot be trained for this resolution due to its expensive memory usage, even with training batch size 1, when using its official code. However, the authors provided a pretrained model for FFHQ-1024 using a progressive training scheme (no released code). This CIPS model has an FID score of 10.07, much worse than ours. This confirms the superiority of our method over its scale-consistent image generation counterpart.

On the MetFaces dataset, CREPS's FID score is 20.52, which is quite close to the score of StyleGAN2-Ada (18.22). As mentioned, CIPS fails to train on this 1024×1024 resolution using its official code. It confirms that bi-line representation does not constrain the adaptability of our model.

On LSUN-Church and Flickr-Scenery, although the unstructured and diverse images in these datasets are intuitively adverse to column and row decomposition, CREPS obtains good results with only a small gap compared with StyleGAN2's ones. Note that CIPS achieves a surprisingly good result on LSUN-Church; it surpasses not only CREPS but also StyleGAN2 in this setting.

Fig. 4 provides some samples synthesized by our networks on the benchmark datasets. As can be seen, CREPS produces highly realistic images in all cases.

4.3. Generate arbitrary-scale images

While our models are trained on images with resolutions from 256×256 to 1024×1024, they can generate images at any scale. One way is that we simply scale the length of e^r and e^c, and the output size is changed accordingly, thanks to our network design. With a V100 GPU (32GB), our models can generate an image up to resolution 3687×3687 in a single run. Or we can generate an image patch-by-patch with suitable coordinate inputs, then combine them together into

22413

Generator	FFHQ-512	FFHQ-1024	LSUN Church-256	MetFaces-1024	Scenery-256
StyleGANv2	3.41	2.84	3.86	18.22*	6.40
CIPS	6.18	10.07 †	2.92	OOM	8.49
CREPS (ours)	4.43	4.09‡	5.50	20.52	7.21

Table 2. Comparison of our method against other works in FID metric. OOM means out-of-memory. '*' means the result is taken from StyleGAN2-Ada paper [12]. '†' means the model is provided without releasing its progressive training code. '‡' means the result is obtained by scaling the output resolution of the FFHQ-512 model.

	PSNR ↑	SSIM ↑	LPIPS ↓
AnyresGAN	24.19	0.73	0.07
ScaleParty	24.50	0.70	0.08
CIPS	33.33	0.93	0.05
CREPS	**34.65**	**0.96**	**0.01**

Table 3. Scale consistency comparison of our method against three other works on PSNR, SSIM and LPIP. The best scores are **bold**.
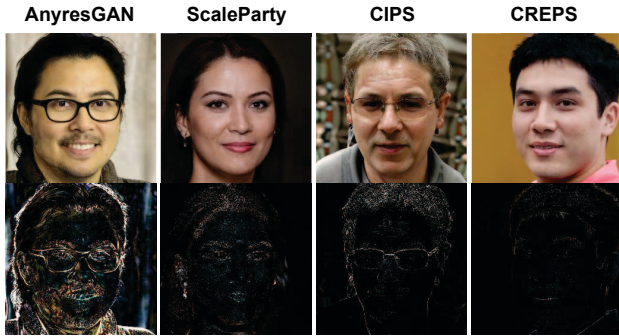
Configuration	FID	Memory	Time
CREP-NB	5.98	2.7GB	0.13s
+ bi-line and d=1	11.37	1.5GB	0.02s
+ bi-line and d=8	8.23	1.6GB	0.03s
+ multiple decoders and d=4	6.46	1.6GB	0.03s
+ multiple decoders and d=8	4.66	1.7GB	0.04s
CIPS	7.08	3.5GB	0.05s

Table 4. Effects of the modifications of CREPS on the FFHQ dataset in terms of FID score, memory usage, and running time.

Figure 5. Qualitative results for the scale consistency experiment. For each method, we provide a sample generated 256×256 image (top) and the magnified ($\times 10$) residual map between it and the 512×512 rescaled version (bottom).

a single gigantic image with no upper limit in the output size. We provide in the Supplementary some images generated at 6K resolution. While these images are not as sharp as real-world ultra-high-resolution images, they are much sharper than the ones generated at resolution 512×512 and then upscaled with Lanczos resampling.

4.4. Image scaling consistency

In this section, we evaluate the scale consistency of images produced by CREPS and other methods, including AnyresGAN [4], ScaleParty [18], and CIPS [1]. We run this experiment using models trained on the FFHQ dataset. For each model, we first randomly generate 10k images at resolution 256×256 (first set). We then generate images with the same latent codes but at resolution 512×512 and downsample them to 256×256 (second set). The images in two sets are expected to be the same. Hence, we can compare two sets, with standard metrics such as PSNR, SSIM, and LPIPS, to measure each model's scale equivariance.

Note that for ScaleParty and AnyresGAN, the pretrained weights are already trained with different resolutions at once, so we directly use their provided version. CIPS, how-

ever, is trained in a single-scale setting, so we use the available weights trained at the highest resolution (1024×1024) but pass the input coordinate with size 512×512 to synthesis image at resolution 512. As for CREPS, we simply use the weight trained at resolution 512×512.

We report the qualitative and quantitative results in Fig. 5 and Tab. 3. As can be seen, it is clear that CREPS achieves the best scale consistency, while convolution-based models like ScaleParty and AnyresGAN perform poorly.

4.5. Ablation studies

To better understand our proposed techniques, we analyze the effect of different parts of CREPS on the FFHQ datasets. We first consider a no-bi-line version of CREPS as a baseline (referred to as *CREPS-NB*), with dense 2D input and the decoder layers removed. We then apply bi-line decomposition but fuse the bi-line features only once at the end, with gradually increased thickness. Lastly, we add multiple decoders for layer-wise feature composition as introduced in Sec. 3.4. Because of limited time and computational resources, we only evaluate on 128×128 resolution and all of our models were trained for maximum of 2 days.

As the results in Tab. 4 show, CIPS performs worst in all three aspects compared with most of our models. Simply adding the bi-line with a single decoder at the end nearly halves the memory costs, but the FID score is still behind CREPS-NB even when the thickness is increased to $d = 8$. However, multiple decoders can help bring the image quality back to the level of CREPS-NB and even better with $d = 8$. In all settings, it can be clearly seen that we easily boost the FID score when increasing the thickness. These observations prove the importance and effectiveness of our proposed techniques. Remarkably, while the decoders seem compute-intensive, they are actually lightweight due to their
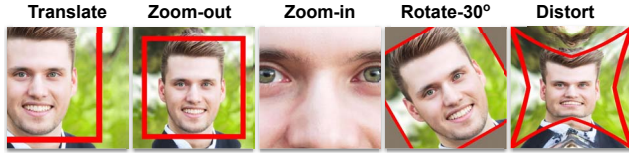
Figure 6. Geometric transformations on the same target image (FFHQ-512 model) by changing the input coordinates. We mark the original image boundary using a red rectangle.
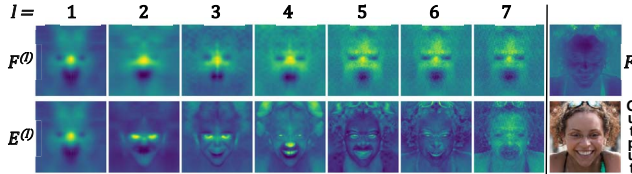

Figure 7. Visualization of the feature maps extracted from our FFHQ-512 model. Each feature map is averaged over all channels.

narrow width compared with other layers, causing only small increases in memory and time.

4.6. Simple and complex geometric transformation

Our CREPS model can support various geometric transformations on the same target image by keeping the input latent code z but changing the input coordinates e^r and e^c. We can translate the image by adding coordinate shifts δ_y and δ_x to the row and column input coordinates, respectively. We can also multiply these input coordinates by the same constant $s > 1$ for zooming out or divide them by s for zooming in. As can be seen in the first three column in Fig. 6, CREPS can perform those simple transformations with consistent details. Notably, CREPS can extrapolate the points outside the original image boundary, although it has never been trained on such input coordinates.

It is tricky for CREPS to handle complex transformations such as rotation or distortion since CREPS only takes in a row and a column coordinate input. Instead of producing the target image in one run, we can execute CREPS multiple times to generate different parts of the output image, then combine them. The simplest algorithm is to sample each target pixel per run by setting a single value for e^r and e^c. However, that algorithm is too slow, which requires 262k runs to produce a single 512×512 image. A faster way is to sample the target image row-by-row. Assuming we need to generate an image I with the input latent z and the target pixels' normalized coordinates $\{(r_{ij}, c_{ij})\}_{i=\overline{1,H}, j=\overline{1,W}}$. We can produce each row I_i of the target image by generating an intermediate image I' using the input coordinates $e^r = [r_{ij}]_{j=\overline{1,W}}$ and $e^c = [c_{ij}]_{j=\overline{1,W}}$ and sample its diagonal $I_i = diag(I')$. We provide two examples with rotation and elastic distortion in the last two columns of Fig. 6. Both images are correctly transformed with unchanged content.

Figure 8. Samples of the most common kinds of artifacts on different datasets. They are best described as repeating/wavy patterns, vertical symmetry, and glowing blobs. Left-most image is cropped and zoomed-in from a full-face image.

4.7. Feature analysis

We visualize the key feature maps inside our FFHQ-512 model when generating a facial image and provide them in Fig. 7. The maps include the layer-wise composed features $\{F^{(l)}\}_{l=\overline{1,N}}$, the corresponding layer-wise fused maps $\{E^{(l)}\}_{l=\overline{1,N}}$, and the final feature map F (see Eq. 4-6). Thanks to the asymmetric fusion scheme, the model seems to synthesize the output in a coarse-to-fine manner. The early composed feature maps are smooth and focus on the global structure, while the later ones focus on sharp details. Although each composed feature map $F^{(l)}$ is quite simple, the network can represent complex content by fusion.

4.8. Limitation

Being a fully-connected generator, CREPS shares the same limitation with other similar work, which is the lack of spatial bias since each pixel is independently generated. Hence, some spatial-related artifacts occasionally occur in our generated images (Fig. 8). A potential cause is the sine activation at the beginning, producing repeating patterns and vertical symmetry of the output. We also note that some samples contain a noticeable blob that is completely out-of-domain. We found CIPS facing the same problem, and the root cause can be the missing spatial guidance from neighboring pixels and the effect of Leaky-RELU activations which strengthens the isolation of some pixel regions.

5. Conclusion

In this paper, we present a new architecture named **CREPS**, a cost-effective and scale-equivariant generator that can synthesize images with any target resolution. Our key contributions are an INR-based design, a thick biline representation, and a layer-wise feature composition scheme. While being more memory-efficient, our CREPS models can produce highly realistic images and surpass the INR-based model CIPS in most cases. CREPS also offers the best scale consistency by keeping image details unchanged when varying the output resolution. We conducted several experiments to explore some attractive properties of this fully-connected generator and discussed CREPS's applications in various scenarios. Future development of our approach can be eliminating artifacts mentioned in Sec. 4.8 and further improving the quality of our samples.

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 7

[2] A.H. Bermano, R. Gal, Y. Alaluf, R. Mokady, Y. Nitzan, O. Tov, O. Patashnik, and D. Cohen-Or. State-of-the-art in the architecture, methods and applications of stylegan. *Computer Graphics Forum*, 41(2):591–611, 2022. 1

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[4] Lucy Chai, Michael Gharbi, Eli Shechtman, Phillip Isola, and Richard Zhang. Any-resolution training for high-resolution image synthesis. In *European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 3, 7

[5] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[6] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3, 4

[7] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. In&out: Diverse image outpainting via gan inversion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1, 2

[9] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations (ICLR)*, 2022. 3

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 2

[11] Animesh Karnewar, Oliver Wang, and Raghu Sesha Iyengar. Msg-gan: Multi-scale gradient gan for stable image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[12] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2, 7

[13] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2, 3

[14] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[15] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 4

[16] Ji Lin, Richard Zhang, Frieder Ganz, Song Han, and Jun-Yan Zhu. Anycost gans for interactive image synthesis and editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[17] 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 (ECCV)*, 2020. 3

[18] Evangelos Ntavelis, Mohamad Shahbazi, Iason Kastanis, Radu Timofte, Martin Danelljan, and Luc Van Gool. Arbitrary-scale image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 1, 2, 3, 7

[19] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 3

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. 2

[21] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 3

[22] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. 3

[23] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3

[24] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2020. 2

[25] 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. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[26] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Nießner, J. T. Barron, G. Wetzstein, M. Zollhöfer, and V. Golyanik. Advances in neural rendering. *Computer Graphics Forum*, 41(2):703–735, 2022. 3

[27] Lance Williams. Pyramidal parametrics. *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, 1983. 3

[28] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022. 1

[29] Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in gans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2