Abstract

Positional encodings have enabled recent works to train a single adversarial network that can generate images of different scales. However, these approaches are either limited to a set of discrete scales or struggle to maintain good perceptual quality at the scales for which the model is not trained explicitly. We propose the design of scale-consistent positional encodings invariant to our generator’s layers transformations. This enables the generation of arbitrary-scale images even at scales unseen during training. Moreover, we incorporate novel inter-scale augmentations into our pipeline and partial generation training to facilitate the synthesis of consistent images at arbitrary scales. Lastly, we show competitive results for a continuum of scales on various commonly used datasets for image synthesis.

1. Introduction

Generative adversarial networks (GANs) [9] are the most commonly used paradigm for generating and manipulating images and videos [12, 24, 25, 29, 30, 36, 37]. The promising results obtained by GANs have motivated several applications of computer graphics and visual content generation. Ideally, a GAN model is not only capable of generating images similar to the training data but also provides the flexibility to manipulate and control the generation process for the target application [13, 27]. For instance, a GAN model used for animations and videos should be able to generate objects in different positions, scales, and viewpoints while maintaining consistency over other attributes of the object. Having a single model that provides control over different object attributes has received substantial attention from the research community [7, 18, 20]. However, most of the existing GAN models are limited to the positional priors of their training data, making them unable to generate unseen translations and scales.

Xu et al. [38] recently revealed that convolutional GANs learn the positional priors of their training data by using the zero paddings in the convolutions as an imperfect and implicit positional encoding. Motivated by such discovery, explicit positional encodings have been proposed to make the GAN models equivariant to different translations, scales,
and resolutions [2,6,33,38]. Positional encodings have created the possibility of obtaining a single GAN model, that can generate images with different resolutions, as well as different object scales and positions. However, despite the new opportunities brought about by the recent works, the existing methods are still limited to multi-scale generation only in discrete resolutions. They suffer from object inconsistency between different scales and resolutions.

To address the aforementioned limitations, we aim to extend the task of multi-scale generation, using a single generator, to arbitrary continuous scales. To this end, we first propose a more suitable positional encoding formulation. While this leads to arbitrary-scale generation, this strategy alone does not guarantee consistency across scales. We therefore further propose a means of enforcing consistency between different scales and resolutions using inter-scale augmentations in the discriminator. Specifically, we generate images at different scales from the same latent code. Then, pairs of generated images at different scales go through channel-mix and cut-mix augmentations. Finally, the discriminator classifies the augmented images as real or fake. Such an approach encourages the generator to generate scale-consistent images so that the images still look realistic after inter-scale augmentations. Lastly, our method can also generate parts of the image in arbitrary resolutions with scale consistency, as visualized in Figure 1.

To summarize our contributions:

- We design a scale-consistent positional encoding scheme that enables fully convolutional and pad-free generators to generate images of arbitrary scales.
- We introduce a set of inter-scale augmentations that pushes the generator to create consistent images among scales.
- We further facilitate the consistency among arbitrary scales by incorporating partial generation in our training pipeline.

We perform experiments on various commonly used datasets characterized by diverse positional priors. Our results indicate that the introduced pipeline permits the consistent generation of images of arbitrary scales while preserving high visual quality.

2. Related Work

Generative adversarial networks have been exploited in various applications for unconditional generation [15], as well as generation constrained by conditions, such as images [42], semantic categories [3,31,32], semantic layouts [24,25], and text [28]. As the main focus of this work, existing methods on partial generation and multi-scale generation based on GANs are discussed in this section.

2.1. Partial Generation

Standard GANs are usually trained to directly map a latent code to a full image. Models capable of partial generation, on the other hand, typically generate different parts of the image independently, which they can then aggregate to construct the full image. As investigated in previous works, partial generation can be posed as both patch-wise [5,21,22,34,43] or pixel-wise [2] generation of the images. The main challenge in partial generation is maintaining the global structure and consistency of the full image. Therefore, position-aware generation using implicit or explicit positional encoding has become a crucial component of partial generation. Positional encodings have also been used in the context of semantic image synthesis [35].

COCO-GAN [21] generates different patches of the image and concatenates them to form the full image. The global consistency is ensured by using a generator that uses positional encodings coupled with a global latent code and a discriminator that assesses the quality of the concatenated patches. Infinity-GAN [22] is another model based on patch generation that combines a local latent code with global latent code and the positional encoding to drive generation. ALIS [34] exploits patch generation to generate images infinitely extendable in the horizontal direction.

INR-GAN [33] and CIPS [2] differ from the aforementioned works as they perform partial generation pixel-wise. Instead of generating image patches using a convolutional network, they exploit fully-connected implicit neural representation (INR) to generate each pixel based on their position in the coordinate grid. The sample-specific parameters of the INR for each image are generated by a hyper-network that receives the latent code as its input.

Contrary to these works, our generator learns global consistency by generating smaller resolution full-frame images and imposing a multi-scale consistency objective.

2.2. Multi-scale Generation

Multi-scale generation can be defined as the task of generating images in different scales using a single model. MSG-GAN [19] can be seen as one of the earlier works on multi-scale generation. Inspired by ProGAN [14], the authors propose an architecture that outputs an RGB image at each layer of the generator, resulting in generating multiple scales of the same image. This approach, however, is only limited to the discrete resolutions up to the resolution of the final output. A recent study called MS-PIE [38], proposes a padding-free fully-convolutional architecture capable of multi-scale generation based on the input positional encoding and the global latent code. The multi-scale generation can be done by feeding different resolutions of the positional encoding to the generator. To avoid shrinkage in the size of the padding-free feature maps, authors use bi-linear upsampling layers that generate feature maps with
extra boundaries, compensating the lack of positional encoding. A similar recent study [6] achieves multi-scale generation by feeding the positional encodings at each layer of the generator, while retain the zero padding. We show that with proper design of positional encodings using them only as input is sufficient for multi-scale generation. Moreover, none of the aforementioned methods tackles the problem of synthesis at arbitrary scales nor addresses whether the multi-scale output is consistent.

CIPS [2] and INR-GAN [33], while trained for a single scale, are able to generate in multiple scale. Note, however, that their single-location conditional input does not contain any information about the scale they aim to generate in.

3. Our Method

We aim to design a generative adversarial network for image synthesis capable of: (a) full-frame or in-parts image generation, (b) generation of arbitrary resolutions, and (c) consistency across different scales and parts.

3.1. An image as a continuous space

By viewing an image in a continuous coordinate space $\mathbb{R}^2$, image generation is seen as sampling image values at discrete locations within a finite rectangular area of this continuous space. We define the scale $s$ of the sampled image as the sampling period, and its resolution $r = (r_x, r_y) \in \mathbb{N}^2$ as the number of sampled points. Accordingly, the dimensions $(w, h)$ of the image in the continuous space are obtained as,

$$(w, h) = (r_x \ast s_x, r_y \ast s_y) \in \mathbb{R}^2. \quad (1)$$

We also need a reference location for the rectangle in the continuous space to specify the rectangular region. We use the image’s center coordinates $c = (c_x, c_y)$. Now, the tuple $a = (c, s, r)$ uniquely describes a sampled image $I_{z,a}$. Therefore, $I_{z,a}[i,j]$—the value for the pixel $(i, j)$ in $I_{z,a}$—is obtained from the continuous image as:

$$I_{z,a}[i,j] = I_z(c_x + s_x i - w/2, c_y + s_y j - h/2) \quad (2)$$

$$I_{z,a}[i,j] = I_z(c_x + s_x i - w/2, c_y + s_y j - h/2) \quad (3)$$

where $z$ is the semantic identifier of the image space. Each different scene/portrait/photograph has each unique $z$.

3.2. Properties of Arbitrary-Scale Synthesis

A convolution-based generator architecture needs specific characteristics to enable arbitrary-scale synthesis in a spatially equivariant and scale-consistent manner. This section will formulate these properties concerning the input positional encodings used as guidance.

**Position-guided generation.** The generator needs to offer the ability to designate where in the image space ($c$) and at which resolution ($r$) and scale ($s$) it should generate the image. We give this information to the network via positional encodings $p_{enc}(a) = p_{enc}(c, r, s)$. Similar to the definition of $I_z$, each element of $p_{enc}$ designates a single location. The sampling period of the locations defines the scale, their number and alignment, the resolution. $p_{enc}$ are different from the latent code $z$, which can be thought of as a description of the scene that produces another image space. Our generator network $G$ maps the latent code $z$ and a positional encoding as to an image space:

$$I_{z,a} = G(z, p_{enc}(a)), \text{ where } a = (c, r, s) \quad (4)$$

**Spatial equivariance** can be formulated as follows: A shift in reference location $c \rightarrow c'$ of the positional encodings should result in a similar shift in the image space,

$$I_{z,a'} = G(z, p_{enc}(a')) \text{ where } a' = (c', r, s) \quad (5)$$

We can similarly define **scale consistency** as equivariance to the scale transformation $s \rightarrow s'$.

$$I_{z,a''} = G(z, p_{enc}(a'')) \text{ where } a'' = (c, r, s') \quad (6)$$

3.3. Designing a scale- and translation-equivariant Generator

We base our generator network $G$ on the commonly used StyleGANv2. First, we discuss the modifications needed to achieve the spatial and scale equivariance. The generator’s architecture is mainly composed of a learned constant input, a modulated convolution layer, and two modulated convolution layers. The convolution layers use zero padding, which keeps the resolution of the input-output feature maps unchanged. The only size-changing operations in the generator are the up-sampling layers. Let the input size be $n_{in} \times n_{in}$. The output resolution is given by:

$$r_L = n_{in} \ast 2^L. \quad (7)$$

This means that the size of the output of a convolutional generator is directly proportional to the size of its input and can only have values with a $2^L$ increment. To synthesize an image where the full-frame resolution is between two consecutive values, *e.g.* $L_1, L_2$, $G$ needs to handle partial synthesis. The end result is either trimmed down $r_{L_2}$ or a stitched up version of smaller outputs.

Both scale and translation equivariance are critical towards our objective. As a generator architecture is a multi-step process, a natural way to impose equations (5) and (6) is for them to hold at each intermediate step. Convolutional layers are, by design, translation equivariant. Thus, we address this property in the rest of the components: padding, upsampling and positional input.
Removing the padding. Zero padding breaks the translation equivariance of the network \([22, 38]\). Removing it strips the network of its positional anchor. Instead, positional encodings guide the generation of the image \([6, 38]\). However, without padding, the 3×3 convolution leads to a shrink-ed output feature map compared to its input.

Pitfalls when upsampling. One approach \([38]\) to counter this shrinkage effect of zero padding, is to change the upsampling operation to a factor larger than two. This provides an excess of pixels in the feature maps that is subsequently consumed by the convolutions. Specifically, as two convolution blocks are applied after the upsampling, the feature maps are resized from \(n_{in} \times n_{in}\) to \(2 \times n_{in} + 4\). However, this approach transforms the space unevenly when applied to different scales. An input of 4×4 will be rescaled with a factor of 3 while an input of 8×8 with a factor of 2.5.

Similarly, an uneven transformation of space happens when resizing is done with aligned corners, both as part of the upsampling operation and the design of input encodings.

Fixed positional corners. Xi et al. \([38]\) argue that using fixed values for the edge positional encodings, same for every scale, provides spatial anchors across the image space. While this is useful for generating images of specific set of scales, it impedes our arbitrary-scale and partial synthesis goal. For translation equivariance, it is crucial for the encodings to point at the center of the pixel and not at the corners. This way, two independently generated patches will be characterized by equally spaced positions. In multiscale synthesis, aligned-corners alter the sampling period between different scales, where \(d_{n \times n} = (w/(n-1), h/(n-1))\).

Alternatively, sampling all the positional encodings at the central location of the patch they produce gives a period of \(d_{n \times n} = (w/n, h/n)\) and thus \(2d_{2n \times 2n} = d_{n \times n}\).

This inter-scale inconsistency of the positional grounding between layers pushes the network to overfit to the scales it is trained to generate. Therefore, the generator is unable to synthesize in a scale in between. We can observe this effect in the first row of Fig. 3.

Scale consistent positional encodings. We address the aforementioned issues in our design of the positional encodings. A grid coordinate system is used as a natural and straightforward way to define them.

As we want the positional encodings to describe the same area as the sampled image \(I_{x,n}\), described by \(a = (c,s,r)\) and the input resolution is \(n \times n\), we find the sampling period to be \(s_{n \times n} = (w/n, h/n)\) as per (1).

To counter the shrinkage effect we utilize feature unfolding \([4,22]\). However, for multiscale synthesis, the unfolding should be used as auxiliary padding and not taken into consideration when designing the encodings’ sampling period to maintain \(2d_{2n \times 2n} = d_{n \times n}\). Therefore, we extrapolate the positional encodings by the constant \(n_{pad}\) on each side. We define the positional encodings as,

\[
p_{enc}(a)[i,j] = \begin{cases} 
(e_x + s_x(i + 0.5) - w/2), \\
(c_y + s_y(j + 0.5) - h/2),
\end{cases}
\forall i, j \in [-n_{pad}, n + n_{pad}) \cap \mathbb{Z}
\]

Note that \(n_{pad}\) does not affect the scale \(s\). Using \(p_{enc}(a)\) as the input to our StyleGAN2-based architecture the resolution of the intermediate feature maps is:

\[
\begin{aligned}
n_{out}^0 &= n + 2n_{pad} - 2 & \text{For the first convolution} \\
n_{out}^l &= n_{out}^{l-1} \ast 2 - 4 & \text{For each upsampling block}
\end{aligned}
\]

By setting \(n_{pad} = 3\) we get:

\[
n_{out}^l = n_{in} \ast 2^l + 4
\]

These extra 4 pixels at the margins of each intermediate feature map are there regardless of the input size. They play the auxiliary role of keeping equation (10) consistent among
layers. We remove them at the end of the network, and thus our output resolution is described by the same formula as its zero-padded counterpart (Equation (7)).

Feature unfolding designates an image area larger than the one we want to generate. The upsampling is doubling the scale without changing the area. The convolutions consume the excess area, but the area described by the initial positional encodings does not change between layers.

A shift in positional encodings translates to a shift of the image. Additionally, changing the spacing between them without increasing their number will change the size of the area they describe and let us generate a continuum of scales.

3.4. Training for scale

While the design choices described in the previous subsection permit the generation of arbitrary-sized images, they do not guarantee consistency among images generated from the same latent code but at different scales. To achieve this, we propose a scale consistency objective.

Training pipeline. In order to train for a multiscale objective, we teach the generator to synthesize images of different scales. For each batch, we randomly choose the output resolutions \( r_{\text{small}} \) and \( r_{\text{large}} = 1.5 \times r_{\text{small}} \) from a predefined set, in accordance with Equation (10).

Assuming a generator with 6 upsampling blocks, we pick \( r_{\text{small}} = 256 \) and \( r_{\text{large}} = 384 \). This gives us \( n_{\text{small}} = 4 \) and \( n_{\text{large}} = 6 \). Then, we randomly choose the scale \( s \) of the image that will be generated and its location \( (c_x, c_y) \).

Lastly, we sample the latent code \( z \). Thus, we get,

\[
I_{z,a_{\text{small}}} = G(z, p_{\text{enc}}(a_{\text{small}})) \tag{11}
\]
\[
I_{z,a_{\text{large}}} = G(z, p_{\text{enc}}(a_{\text{large}})) \tag{12}
\]

Similarly, we crop and resize the real images per \( (c_x, c_y) \).

Scale consistency. The classic adversarial training only pushes images to look realistic at each scale. We need to define an objective that will teach the generator to match the outputs. A straightforward approach is to impose a distance metric, such as L1 loss, between images generated at different scales and subsequently resized to match. However, this can give the network conflicting incentives. The L1 loss drives the different images to match without any regard to their perceptual quality; two uniformly black images would achieve the perfect L1 loss.

We propose a scale consistency approach that strives simultaneously to generate similar images at different scales and images that look realistic. To achieve this, we use augmentation techniques during the training of the discriminator without changing its loss function.

We deploy two types of augmentations before feeding \( I_{z,a_{\text{small}}} \) and \( I_{z,a_{\text{large}}} \) to the discriminator. First, we use CutMix [40] to crop a region at one scale and substitute it with a resized crop of the same region of the image generated at the other scale. Then, we use ChannelMix to randomly substitute some of the RGB channels of the image at each scale with ones from its counterpart, after it is resized to match.

The discriminator is trying to measure the realness of the mixed images. In the process, the generator learns to associate the identity of images it synthesizes with the style code and their position and scale with the input positional map. The whole pipeline of our method is shown in Fig. 2.

Global consistency for partial generation by Multiscale Training. Combining partial and multiscale training naturally counters a common partial synthesis problem: global consistency. The generator can create a consistent large resolution full-frame image at inference, without explicitly trained for it. The network learns the global structure by being taught to generate small-resolution full-frame images, and detailed textures of high-resolution patches.

Handling Injected Noise during inference. In Equation (4) we described a simplified formulation of the generator that omitted the injected noise at the end of each convolution. We strive for consistency among different scales of images produced with the same latent code, but randomly sampling the injected noise works against this objective.

Imposing scale-consistent positional encodings enables a practical feature. We know the positional grounding of every pixel of every intermediate feature map. This lets us have a position-aware interpolation of the noise to match corresponding pixels between scales.

Similarly, the same technique can be used towards translation equivariant synthesis. We shift the intermediate noise
Figure 4. FID scores for entire face generation for a continuum of scales for the FFHQ dataset. The continuous lines indicate the methods that can generate in arbitrary scales. ScaleParty performs competitively to single-scale models.

Table 1. Self-SSIM between 5k FFHQ generated images of different scales, resized and compared at resolution 256×256.

| Method                | Self-SSIM(5k) |
|-----------------------|---------------|
|                       | 320 | 384 | 448 | 512 |
| MSPIE [38]            | 0.1194 0.5929 0.3316 0.5785 |
| MS-PE [6]             | 0.9128 0.8687 0.8367 0.8112 |
| CIPS-256 [2]          | 0.9991 0.9987 0.9985 0.9981 |
| ScaleParty-noSC/Full  | 0.7154 0.6975 0.6489 0.6511 |
| ScaleParty-Full       | 0.8637 0.8942 0.8266 0.8114 |
| ScaleParty            | 0.8802 0.8779 0.8568 0.8454 |

Table 2. Evaluation Metrics on LSUN Church and Bedroom datasets [39]. The datasets do not exhibit strong positional prior, which increases the performance gain of our approach.

| Dataset       | Method | Res | FID | Prec | Rec | SelfSSIM(5k) |
|---------------|--------|-----|-----|------|-----|--------------|
| LSUN Church   | MSPIE  [38] | 128 | 6.67 | 71.95 | 44.59 | 1.00 | 0.32 | 0.43 |
|               |        | 160 | 10.76 | 66.21 | 36.95 | 0.31 | 1.0  | 0.40 |
|               | ScaleParty-noSC/Full | 192 | 6.02 | 66.70 | 46.16 | 0.39 | 0.38 | 1.00 |
| LSUN Bedroom  | MSPIE  [38] | 128 | 7.62 | 70.21 | 39.84 | 1.00 | 0.58 | 0.49 |
|               |        | 160 | 4.74 | 72.23 | 39.44 | 0.55 | 1.00 | 0.67 |
|               | ScaleParty-noSC/Full | 192 | 7.40 | 67.83 | 39.93 | 0.44 | 0.64 | 1.00 |
|               | ScaleParty | 128 | 9.08 | 70.52 | 32.10 | 1.00 | 0.95 | 0.93 |
|               |        | 160 | 7.96 | 70.87 | 32.07 | 0.94 | 1.00 | 0.95 |
|               |        | 192 | 7.52 | 68.14 | 33.33 | 0.90 | 0.94 | 1.00 |

4. Experimental Results

4.1. Implementation

We base our implementation on MS-PIE [38] using the mmgenneration framework [8] built upon PyTorch [26]. For all upsampling operations, we use bilinear interpolation without corner alignment. In order to match the feature maps’ marginal pixels after upsampling, our model is trained with the non-saturating logistic loss, with R1 gradient penalty [23] for the discriminator and path regularization for the generator [16]. We used the StyleGAN2 discriminator [16] together with an adaptive average pooling layer before the last linear layer [10,38]. For all our experiments we set $h = w = 2$ for the encodings calculation.

4.2. Evaluation

Datasets. We evaluate using three different datasets:

- The Flick-Faces-HQ (FFHQ) [15] is composed of 70,000 images of diverse human faces. This dataset is characterized by a strong positional prior as the images are cropped and aligned from photographs with a larger context, based on facial landmarks. The original size of the pictures is 1024×1024. We train on FFHQ by cropping and then downsampling the images.

- The LSUN dataset [39] consists of images that are resized, so their smaller side is 256 pixels. We test our method in two subcategories of the dataset: the LSUN Bedroom, which consists of 3 million bedroom images and the LSUN Church, which has 126 thousand diverse outdoor photographs of churches. While each dataset depicts a similar layout of bedroom and outdoor churches scene, the positional priors of the images are not as strong as in FFHQ. To further reduce their strength we randomly crop square patches of the images while training, without altering the aspect ratio.

Metrics. We rely on commonly used metrics to measure two aspects of multiscale-generation. Frechet Inception Distance [11] assesses the perceptual quality at each scale. It is shown to align with human subjects’ perceptual judgement of an image. Improved Precision and Recall [17] is used to gauge the plausibility of the synthesized images and how well these images cover the range of the distribution of the real images, respectively. To assess the consistency among images generated at different scales, we deploy the SSIM metric. We call it SelfSSIM.

Note that consistency on its own should not be the goal: two equally bad syntheses can have high fidelity among them. SelfSSIM is used with FID to assess whether the generated images are perceptually good and consistent.

4.3. Quantitative Results

Comparison with state-of-the-art models on FFHQ. We use the FFHQ dataset to conduct a comparative analysis with state-of-the-art methods in multiscale generation.

1Code: https://github.com/vglsd/ScaleParty
We test against methods designed for multiscale synthesis: MSPIE [38] and MS-PE [6]. From the INR-based methods, we compare against CIPS [2] as it reports better FID than INR-GAN [33] and their implementation readily handles synthesis at arbitrary scales. We report the results of two models: one trained for 256×256 and one trained for 1024×1024 images. Lastly, we include the instances of the single-scale StyleGAN2 [16] model as a benchmark.

In Fig. 4 and Table 1 we are reporting the FID scores and the SelfSSIM scores respectively. To calculate both metrics we did not use the truncation trick.

ScaleParty vs. other methods. Only StyleGAN & MSPIE consistently yield better FID scores than our approach. However, they overfit the set of scales they were trained for and are incapable of good syntheses outside this set. CIPS has a competitive score for the single scale it was trained for, which rapidly deteriorates as we move away from that scale. CIPS has the best SelfSSIM. Note, CIPS is conditioned on a single location that does not contain any scale information. Generating in higher scales could emulate a naive upsampling method, which similarly would yield almost perfect SelfSSIM. Therefore, ScaleParty is the only method that can consistently achieve low FID scores while maintaining high inter-scale consistency.

The effects of ScaleParty components on FFHQ. We train and compare with two versions of our model, abating on our proposed elements: (a) ScaleParty-noSC/Full is trained with our proposed scale-invariant positional encodings, but only with full-frame images of a discrete set of scales and no consistency objective. (b) ScaleParty-Full is trained with full-frame images and an additional scale-consistency objective: in 20% of the batches, we generate a multiscale pair of images. In contrast, our full model ScaleParty is trained with both the scale-consistency objective and for partial generation. During training, the positional encodings (and real images respectively) are sampled with a scale 60 − 110% of the full-frame.

We find that increasing the inter-scale consistency comes at a slight drop in perceptual quality. As seen in Fig. 4, ScaleParty-noSC/Full produces the best FID score compared to the configurations imposing scale consistency. Partial generation trains the generator for different scales. While ScaleParty-Full results in better SelfSSIM for the trained full image resolutions, we observe a drop in consistency for the scales the network was not trained with. However, upon visual inspection we notice an unnatural distortion in the faces generated without partial synthesis training, that is not reflected in the FID, as seen in Figure 4 of the supplementary material for both ScaleParty-noSC/Full and ScaleParty-Full. This distortion explains the lower SelfSSIM between the unseen and the trained-for scales.

The effects of ScaleParty components on LSUN Dataset. In contrast to FFHQ, LSUN lacks strong positional priors. The difference is intensified due to the random cropping. For investigating this setting we train MSPIE as our baseline, as it also deploys a pad-free generator. Furthermore we train ScaleParty-noSC/Full along with our main configuration, ScaleParty, to illustrate the benefits of our scale-invariant design and our scale consistency objectives respectively. In Table 2 we can see the results.

MSPIE and ScaleParty-noSC/Full are trained with 128×128 and 192×192 full-frame images, sampled with equal chance. The inconsistency between the positional encodings hinders MSPIE’s association of the positional input to the output. Both noise injection and generation at different scales lead to a change of the location of the generated images, resulting in a poor SelfSSIM, even with good FID. In contrast, our positional encodings learn the association, enabling good synthesis at the unseen resolution of 160.

Compared to the positionally structured FFHQ where MSPIE and ScaleParty-noSC/Full achieve relatively high SelfSSIM, these configuration exhibit poor consistency in LSUN datasets. In contrast, our ScaleParty shows similarly high results, recording even higher increase compared to the face dataset. We refer to the supplementary material for visual comparisons on both FFHQ and LSUN datasets.

Ablation on scale consistency approaches. We investigate the effect scale consistency has on the perceptual quality of the generated images. We start with our ScaleParty-noSC/Full model, where no such objective is imposed. We experiment applying L1 loss and combinations of our sug-
4.4. Applications

Geometric manipulation using positional encodings. Training for both multiscale and partial synthesis requires the convolutional generator to learn to interpret a great variety of positional encodings configurations. We present it with unseen configurations to test how well the generator learned to translate the positional input. We show qualitative results of applying transformation on the input positional encodings. In Fig. 6 and Fig. 1c we can observe: (a) transformation of the aspect ratio, (b) warping, (c) unseen resolutions and (d) extrapolation. Projection of real images. We investigate the ability of our network to represent real images within its latent space. Following Abdal et al. [1] we deploy optimization of the style vectors modulating each layer of our network ($W^+_{space}$). We aim to minimize the perceptual [41] and $L_2$ loss between the real and generated images while keeping the weights of the generator frozen.

We find that optimizing the latent code for a single-scale image leads to scale overfitting. However, by optimizing the same latent code for two scales simultaneously we are able to also generate in all scales in between. In Fig. 7 we use the repertoire of transformations described in the previous subsection to geometrically manipulate a real images.

5. Conclusion

We present ScaleParty, a novel method for Arbitrary-Scale Image Synthesis utilizing a single generative adversarial network trained with positional guidance. We show that our scale-consistent positional encodings permit a pad-free generator to produce perceptually good results across a continuum of scales. Furthermore, we introduce a scale-consistency objective by applying inter-scale augmentations before presenting the synthesized image to the discriminator network. Incorporating partial generation training in our pipeline further improves consistency. The combination of multi-scale and partial synthesis training teaches the generator a dense representation of positional encodings. During inference, this can be leveraged to create geometrically manipulated images by applying transformations such as warping or stretching to positional encodings.

Acknowledgements This work was partly supported by CSEM and the ETH Future Computing Laboratory (EFCL), financed by a gift from Huawei Technologies.
References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 4431–4440, 2019.

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

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In Proceedings of the International Conference on Learning Representations (ICLR), 2019.

[4] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8628–8638, 2021.

[5] 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, 2021.

[6] Jooyoung Choi, Junghoem Lee, Yonghyun Jeong, and Sungroh Yoon. Toward spatially unbiased generative models. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 14253–14262, October 2021.

[7] Edo Collins, Raja Bala, Bob Price, and Sabine Süsstrunk. Editing in style: Uncovering the local semantics of gans. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5770–5779, 2020.

[8] MMGeneration Contributors. MMGeneration: Openmmlab generative model toolbox and benchmark. https://github.com/open-mmlab/mmgeneration, 2021.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, pages 2672–2680, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, pages 346–361, Cham, 2014. Springer International Publishing.

[11] 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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, pages 6626–6637, 2017.

[12] Seunghoon Hong, Xinchen Yan, Thomas E Huang, and Honglak Lee. Learning hierarchical semantic image manipulation through structured representations. In Advances in Neural Information Processing Systems, pages 2713–2723, 2018.

[13] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks, 2020.

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In Proceedings of the International Conference on Learning Representations (ICLR), 2018.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[16] Tero Karras, Samuli Laine, Miikka Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In Proc. CVPR, 2020.

[17] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. CoRR, abs/1904.06991, 2019.

[18] Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Wald, Gal Elidan, Avinatan Hassidim, William T. Freeman, Phillip Isola, Amir Globerson, Michal Irani, and Inbar Mosserti. Explaining in style: Training a gan to explain a classifier in stylespace. arXiv preprint arXiv:2104.13369, 2021.

[19] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. arXiv preprint arXiv:1907.11922, 2019.

[20] Bingchuan Li, Shaofei Cai, Wei Liu, Peng Zhang, Miao Hua, Qian He, and Zili Yi. Dystyle: Dynamic neural network for multi-attribute-conditioned style editing, 2021.

[21] Chieh Hubert Lin, Chia-Chi Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. Cocogan: Generation by parts via conditional coordinating. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Oct 2019.

[22] Chieh Hubert Lin, Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, and Ming-Hsuan Yang. Infinitygan: Towards infinite-resolution image synthesis. arXiv preprint arXiv:2104.03963, 2021.

[23] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 3481–3490. PMLR, 10–15 Jul 2018.

[24] Evangelos Ntavelis, Andrés Romero, Iason Kastanis, Luc Ward, Tiberiu Serban, and Ming-Hsuan Yang. Infinitygan: To-wards infinite-resolution image synthesis. arXiv preprint arXiv:2104.03963, 2021.

[25] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner,
Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alchê-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

[27] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylenet imagery, 2021.

[28] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR.

[29] Andrés Romero, Pablo Arbeláez, Luc Van Gool, and Radu Timofte. Smit: Stochastic multi-label image-to-image translation. In Proceedings of the International Conference on Computer Vision (ICCV), Workshops, 2019.

[30] Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In International Conference on Learning Representations, 2021.

[31] Mohamad Shahbazi, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Collapse by conditioning: Training class-conditional GANs with limited data. In International Conference on Learning Representations, 2022.

[32] Mohamad Shahbazi, Zhiwu Huang, Danda Pani Paudel, Ajad Chhatkuli, and Luc Van Gool. Efficient conditional gan transfer with knowledge propagation across classes. In 2021 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, 2021.

[33] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. arXiv preprint arXiv:2011.12026, 2020.

[34] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable, 2021.

[35] Zhentao Tan, Dongdong Chen, Qi Chu, Menglei Chai, Jing Liao, Mingming He, Lu Yuan, Gang Hua, and Nenghai Yu. Efficient semantic image synthesis via class-adaptive normalization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[36] Evangelos Ververas and Stefanos Zafeiriou. Slidergan: Synthesizing expressive face images by sliding 3d blendshape parameters, 2019.

[37] Yuri Viazovetskyi, Vladimir Ivashkin, and Evgeny Kashin. Stylegan2 distillation for feed-forward image manipulation. arXiv preprint arXiv:2003.03581, 2020.

[38] Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in gans. In arxiv, December 2020.

[39] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365, 2015.

[40] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.

[41] 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.

[42] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Computer Vision (ICCV), 2017 IEEE International Conference on, 2017.

[43] Łukasz Struski, Szymon Knop, Jacek Tabor, Wiktor Daniec, and Przemyslaw Spurek. Locogan – locally convolutional gan, 2020.
1. Societal Impact

The growth of deepfakes appearing online is a cause for serious concern in a multitude of domains including: politics, non consensual usage of data and the general feeling of losing faith in digital information. GANs are the main technological advancement that enabled the rise of this content. The presented work does not directly lend itself to creation of fake material, in the sense of replacing faces or creating facial expressions based on audio stream. Indirectly though, our method can be used to geometrically manipulate images and in this sense provide malevolent users an additional tool. Efforts in both the US (S. Rept. 116-289 - IDENTIFYING OUTPUTS OF GENERATIVE ADVERSARIAL NETWORKS ACT) and the EU(2021/0106(COD) Artificial Intelligence Act) are aiming to legislate the creation of deepfakes, while private companies try to detect and ban the spreading of deepfake material on the internet. Our method gravitates towards white colored faces in the center of the latent space due to the imbalance on the used data set. There is a clear need to create diverse data sets, where people are represented equally independent of their ability to access technological resources. This will enable research to be used in a more wide spectrum of applications across the globe. In terms of the ever increasing computational costs of training deep neural networks, our presented method overcomes the need for creating independent models at each resolution. It can be therefore be used to reduce the required energy by replacing multiple single resolution models with a single scale consistent one.

2. Limitations

We use artificially multi-scale datasets to train ScaleParty. We downsample the images to acquire different scales. Parmar et al. [4] argue that different resizing libraries and methods can have drastic effects on the quality of the resized images. This is an aspect we have not investigated. In equation (5) of the main paper we assume that transitive closure applies to resizing, e.g. resizing from 512×512 to 256×256 is equivalent to resizing to 384×384 as an intermediate step. While this assumption is not true, it still helps us with our scale-consistency objective. Nevertheless, an analysis on a naturally multi-scale dataset would greatly benefit the conclusions of this work.
3. The generator’s architecture

In Fig. 2 we can see a schematic of ScaleParty’s generator.

4. Multi-scale training policies

In this section we discuss the different scale training policies for FFHQ [3] that we and the methods we compare with deploy:

- CIPS [1] is trained with one target scale: 256.
- MS-PE [2] is trained for 256, 320, 384, 448, 512.
- MSPIE [5] is trained for 256, 384, 512. We use the version of MSPIE with cartesian spatial grid encodings, as it performs the best in terms of FID. Other encoding configurations exhibit similar behavior in the scales they were not trained for.
- We deploy the same setting as MSPIE for ScaleParty-noSC/Full and train for 256, 384 and 512.
- Our ScaleParty-Full which is trained with the scale consistency objective is trained with output resolutions of 256 and 384, but it can perform well even in higher resolutions.
- Our ScaleParty is also trained with output resolutions of 256 and 384. However, in contrast to all aforementioned approaches this is trained for partial generation; the generator is tasked to synthesized a multitude of scales. For example, during training it is generating 384 pixel parts of a 512 resolution full face picture.

For LSUN Datasets [6], we trained MSPIE, ScaleParty-noSC/Full and ScaleParty with outputs of 128 and 192.

In order to facilitate faster and efficient training, we train our scale consistent versions of ScaleParty by continuing from an earlier checkpoint of the ScaleParty-noSC/Full version.

5. Visual Results

In this section we show a qualitative comparison between the state-of-the-art methods and different versions of ScaleParty.

**FFHQ [3]:** In Fig. 3 we can see visual results of the pre-trained models of CIPS [1], MS-PE [2] and [5]. In Fig. 4 we can see the results for ScaleParty-noSC/Full, ScaleParty-Full and ScaleParty. While FID is lower for most scales for the versions trained with only full images, we can observe that the network applies a peculiar effect on the eyes of the faces it generates in scales it did not train for. We can see that both applying the scale consistency objective and partial generation is important for achieving consistent synthesis in arbitrary scales.

Moreover, in Fig. 1 we can see images synthesized at arbitrary scales. As the generator can only output certain resolutions, for scales between them, we generate at a higher resolution and crop the relevant part.

**LSUN [6]:** In Fig. 5 we can see the qualitative results of MSPIE [5], ScaleParty-noSC/Full and ScaleParty trained for LSUN Bedroom and Church datasets. Note, that in combination with the weaker positional prior that these datasets have compared to FFHQ, we further augment this disparity by applying random cropping as a preprocessing step. Compared with FFHQ, MSPIE is generating more coherent results in the intermediate scale. However, in the case of LSUN Bedroom we can observe that the results are not consistent among different scales.

In Fig. 5, we visualize multiple syntheses of the same latent code and scale but with resampling the injected noise. We observe that ScaleParty is the most consistent among runs, while for MSPIE trained for LSUN Bedroom, we see that noise affects the generated images structurally.

6. UI tool for guided generation

We developed an interactive graphical user interface that permits the user to change the location, zoom factor, size of
Figure 3. Qualitative results of state-of-the-art methods on FFHQ [3]. All images were picked randomly and generated without using the truncation trick. We find that the generated results from CIPS [1] and MS-PE [2] exhibit a lot more artifacts than MS-PIE and our methods. However, note that MSPIE, while it performs the best in terms of FID among all methods, it is unable to generate in scales it was not trained for and it is the least consistent between the scales it generates.

Figure 4. Qualitative results of different versions of our method on FFHQ [3]. All images were picked randomly and generated without using the truncation trick. We note that by drawing more samples the amount of generated images by our models that exhibit visual artifacts is comparable with MSPIE [5] for the scales it was trained for, as it is also supported in the FID calculation. For our methods, we observe that only ScaleParty is able to generate results that are consistent, even for arbitrary scales.

input, aspect ratio and warping of the positional encodings to guide the generation. Please refer to the accompanied video for more details. The tool will be available along with the code and the pretrained models.
Figure 5. Qualitative results on LSUN Bedroom and Church datasets. All images were picked randomly and generated without using the truncation trick.

In Fig. 7 we showcase how changing the position, number and layout of the positional encodings can affect the generated image.

References

[1] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. arXiv preprint arXiv:2011.13775, 2020. 2, 3

[2] Jooyoung Choi, Jungbeom Lee, Yonghyun Jeong, and Sun-
Figure 7. Using our tool, we generate various images using the same latent code. The generated images are connected on their upper left corner with the positional encodings used to guide them. Changing the layout of the input yields different scales, resolutions and transformations. The gray box indicates the area the full face should occupy. The green dots show the actual area of the image space that is generated, while the red ones indicate the positional padding of the input, we utilize to counter the shrinking effect of padding-free convolutions.

groh Yoon. Toward spatially unbiased generative models. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 14253–14262, October 2021.

[3] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[4] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On buggy resizing libraries and surprising subtleties in fid calculation. arXiv preprint arXiv:2104.11222, 2021.

[5] Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in gans. In arxiv, December 2020.

[6] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365, 2015.