TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up

Yifan Jiang[†], **Shiyu Chang**[‡], **Zhangyang Wang**[†]
[†]Department of Electrical and Computer Engineering, University of Texas at Austin
[‡]MIT-IBM Watson AI Lab
[†]{yifanjiang97,atlaswang}@utexas.edu, [‡]shiyu.chang@ibm.com

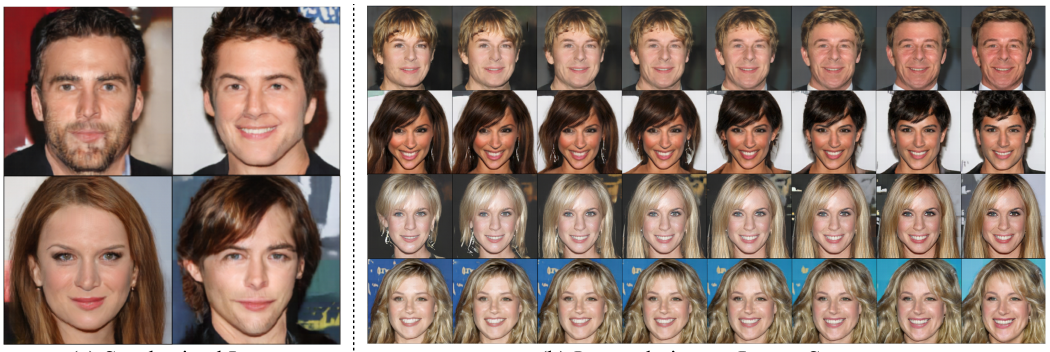
Abstract

The recent explosive interest on transformers has suggested their potential to become powerful "universal" models for computer vision tasks, such as classification, detection, and segmentation. While those attempts mainly study the discriminative models, we explore transformers on some more notoriously difficult vision tasks, e.g., generative adversarial networks (GANs). Our goal is to conduct the first pilot study in building a GAN *completely free of convolutions*, using only pure transformer-based architectures. Our vanilla GAN architecture, dubbed **TransGAN**, consists of a memory-friendly transformer-based generator that progressively increases feature resolution, and correspondingly a multi-scale discriminator to capture simultaneously semantic contexts and low-level textures. On top of them, we introduce the new module of grid self-attention for alleviating the memory bottleneck further, in order to scale up TransGAN to high-resolution generation. We also develop a unique training recipe including a series of techniques that can mitigate the training instability issues of TransGAN, such as data augmentation, modified normalization, and relative position encoding. Our best architecture achieves highly competitive performance compared to current state-of-the-art GANs using convolutional backbones. Specifically, TransGAN sets **new state-of-the-art** inception score of 10.43 and FID of 18.28 on STL-10. It also reaches the inception score of 9.02 and FID of 9.26 on CIFAR-10, and 5.28 FID on CelebA $\mathbf{128} \times \mathbf{128}$, respectively: both on par with the current best results and outperforming StyleGAN-V2. When it comes to higher-resolution (e.g. $\mathbf{256} \times \mathbf{256}$) generation tasks, such as on CelebA-HQ and LSUN-Church, TransGAN continues to produce diverse visual examples with high fidelity and impressive texture details. In addition, we dive deep into the transformer-based generation models to understand how their behaviors differ from convolutional ones, by visualizing training dynamics. The code is available at `https://github.com/VITA-Group/TransGAN`.

1 Introduction

Generative adversarial networks (GANs) have gained considerable success on numerous tasks [1–7]. Unfortunately, GANs suffer from the notorious training instability, and numerous efforts have been devoted to stabilizing GAN training, introducing various regularization terms [8–11], better losses [1, 12–14], and training recipes [15, 16]. Among them, one important route to improving GANs examines their *neural architectures*. [17, 8] reported a large-scale study of GANs and observed that when serving as (generator) backbones, popular neural architectures perform comparably well across the considered datasets. Their ablation study suggested that most of the variations applied in the ResNet family resulted in very marginal improvements. Nevertheless, neural architecture search (NAS) was later introduced to GANs and suggests enhanced backbone designs are also important for improving GANs, just like for other computer vision tasks. Those works are consistently able to discover stronger GAN architectures beyond the standard ResNet topology [18–20]. Other efforts

Preprint. Under review.



(a) Synthesized Image (b) Interpolation on Latent Space

Figure 1: Representative visual examples synthesized by TransGAN, **without using a single convolution**. (a) The synthesized visual examples on CelebA-HQ (256×256) dataset. (b) The linear interpolation results between two latent vectors, on CelebA-HQ (256×256) dataset.

include customized modules such as self-attention [21], style-based generator [22], and autoregressive transformer-based part composition [23].

However, one last "commonsense" seems to have seldomly been challenged: using convolutional neural networks (CNNs) as GAN backbones. The original GAN [24, 25] used fully-connected networks and can only generate small images. DCGAN [26] was the first to scale up GANs using CNN architectures, which allowed for stable training for higher resolution and deeper generative models. Since then, in the computer vision domain, every successful GAN relies on CNN-based generators and discriminators. Convolutions, with the strong inductive bias for natural images, crucially contribute to the appealing visual results and rich diversity achieved by modern GANs.

Can we build a strong GAN completely free of convolutions? This is a question not only arising from intellectual curiosity, but also of practical relevance. Fundamentally, a convolution operator has a local receptive field, and hence CNNs cannot process long-range dependencies unless passing through a sufficient number of layers. However, that is inefficient, and could cause the loss of feature resolution and fine details, in addition to the difficulty of optimization. Vanilla CNN-based models are therefore inherently not well suited for capturing an input image's "global" statistics, as demonstrated by the benefits from adopting self-attention [21] and non-local [27] operations in computer vision. Moreover, the spatial invariance possessed by convolution poses a bottleneck on its ability of adapting to spatially varying/heterogeneous visual patterns, which also motivates the success of relational network [28], dynamic filters [29, 30] and kernel prediction [31] methods.

1.1 Our Contributions

This paper aims to be the **first pilot study** to build a GAN completely free of convolutions, using only pure transformer-based architectures. We are inspired by the recent success of transformer architectures in computer vision [32–34]. Compared to parallel generative modeling works [21, 23, 35] that applied self-attention or transformer encoder in conjunction with CNN-based backbones, our goal is more ambitious and faces several daunting gaps ahead. First and foremost, although a pure transformer architecture applied directly to sequences of image patches can perform very well on image classification tasks [34], it is unclear whether the same way remains effective in generating images, which crucially demands the spatial coherency in structure, color, and texture, as well as the richness of fine details. The handful of existing transformers that output images have unanimously leveraged convolutional part encoders [23] or feature extractors [36, 37]. Moreover, even given well-designed CNN-based architectures, training GANs is notoriously unstable and prone to mode collapse [15]. Training vision transformers are also known to be tedious, heavy, and data-hungry [34]. Combining the two will undoubtedly amplify the challenges of training.

In view of those challenges, this paper presents a coherent set of efforts and innovations towards building the **pure** transformer-based GAN architectures, dubbed **TransGAN**. A naive option may directly stack multiple transformer blocks from raw pixel inputs, but that would scale poorly due to memory explosion. Instead, we start with a memory-friendly transformer-based generator by gradually increasing the feature map resolution in each stage. Correspondingly, we also improve the discriminator with a multi-scale structure that takes patches of varied size as inputs, which

balances between capturing global contexts and local details, in addition to enhancing memory efficiency more. Based on the above generator-discriminator design, we introduce a new module called *grid self-attention*, that alleviates the memory bottleneck further when scaling up TransGAN to high-resolution generation (e.g. 256×256).

To address the aforementioned instability issue brought by both GAN and Transformer, we also develop a unique training recipe in association with our innovative TransGAN architecture, that effectively stabilizes its optimization and generalization. That includes showings the necessity of data augmentation, modifying layer normalization, and replacing absolute token locations with relative position encoding. Our contributions are outlined below:

- **Novel Architecture Design:** We build the first GAN using purely transformers and no convolution. TransGAN has customized a memory-friendly generator and a multi-scale discriminator, and is further equipped with a new grid self-attention mechanism. Those architectural components are thoughtfully designed to balance memory efficiency, global feature statistics, and local fine details with spatial variances.
- **New Training Recipe:** We study a number of techniques to train TransGAN better, including leveraging data augmentation, modifying layer normalization, and adopting relative position encoding, for both generator and discriminator. Extensive ablation studies, discussions, and insights are presented.
- **Performance and Scalability:** TransGAN achieves highly competitive performance compared to current state-of-the-art GANs. Specifically, it sets new state-of-the-art inception score of 10.43 and FID score of 18.28 on STL-10. It also reaches competitive 9.02 inception score and 9.26 FID on CIFAR-10, and 5.28 FID score on CelebA 128×128, respectively. Meanwhile, we also evaluate TransGAN on higher-resolution (e.g., 256×256) generation tasks, where TransGAN continues to yield diverse and impressive visual examples.

2 Related Works

Generative Adversarial Networks. After its origin, GANs quickly embraced fully convolutional backbones [26], and inherited most successful designs from CNNs such as batch normalization, pooling, ReLU/Leaky ReLU and more [38–40, 18]. GANs are widely adopted in image-to-image translation [3, 4], image enhancement [7, 41, 42], and image editing [43, 44]. To alleviate its unstable training, a number of techniques have been studied, including the Wasserstein loss [45], the style-based generator [22], progressive training [16], and Spectual Normalization [46].

Transformers in Computer Vision. The original transformer was built for NLP [47], where the multi-head self-attention and feed-forward MLP layer are stacked to capture the long-term correlation between words. A recent work [34] implements highly competitive ImageNet classification using pure transformers, by treating an image as a sequence of 16×16 visual words. It has strong representation capability and is free of human-defined inductive bias. In comparison, CNNs exhibit a strong bias towards feature locality, as well as spatial invariance due to sharing filter weights across all locations. However, the success of original vision transformer relies on pretraining on large-scale external data. [48, 49] improve the data efficiency and address the difficulty of optimizing deeper models. Other works introduce the pyramid/hierarchical structure to transformer [50–52] or combine it with convolutional layers [53, 54]. Besides image classification task, transformer and its variants are also explored on image processing [37], point cloud [55], semantic segmentation [56], object detection [32, 57] and so on. A comprehensive review is referred to [58].

Transformer Modules for Image Generation. There exist several related works combining the transformer modules into image generation models, by replacing certain components of CNNs. [59] firstly formulated image generation as autoregressive sequence generation, for which they adopted a transformer architecture. [60] propose sparse factorization of the attention matrix to reduce its complexity. While those two works did not tackle the GANs, one recent (concurrent) work [23] used a convolutional GAN to learn a codebook of context-rich visual parts, whose composition is subsequently modeled with an autoregressive transformer architecture.The authors demonstrated success in synthesizing high-resolution images. However, the overall CNN architecture remains in place (including CNN encoder/decoder for the generators, and a fully CNN-based discriminator), and the customized designs (e.g, codebook and quantization) also limit their model's versatility. Another concurrent work [35] employs a bipartite self-attention on StyleGAN and thus it can propagate latent variables to the evolving visual features, yet its main structure is still convolutional, including both the

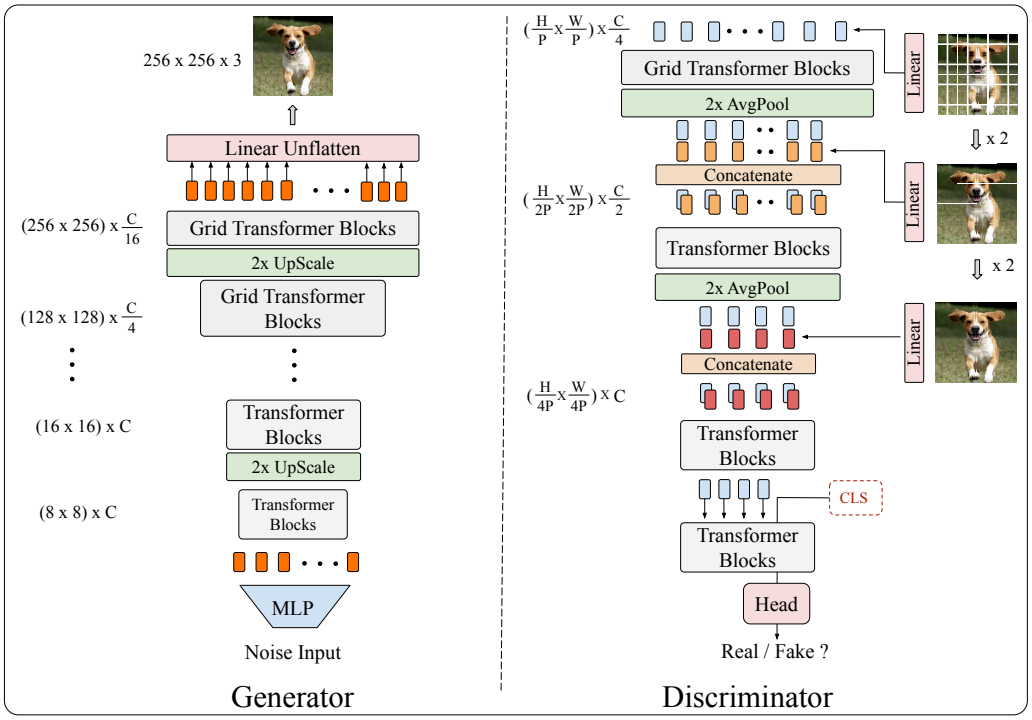


Figure 2: The pipeline of the pure transform-based generator and discriminator of TransGAN. We take 256×256 resolution image generation task as a typical example to illustrate the main procedure. Here patch size p is set to 32 as an example for the convenience of illustration, while practically the patch size is normally set to be no more than 8×8, depending on the specific dataset. `Grid Transformer Blocks` refers to the transformer blocks with the proposed grid self-attention. Detailed architecture configurations are included in Appendix C.

generator and discriminator. To our best knowledge, no other existing work has tried to completely remove convolutions from their generative modeling frameworks.

3 Technical Approach: A Journey Towards GAN with Pure Transformers

In this section, we start by introducing the memory-friendly generator and multi-scale discriminator, equipped with a novel grid self-attention. We then introduce a series of training techniques to stabilize its training procedure, including data augmentation, the modified normalization, and injecting relative position encoding to self-attention.

To start with, we choose the transformer encoder [47] as our basic block and try to make minimal changes. An encoder is a composition of two parts. The first part is constructed by a multi-head self-attention module and the second part is a feed-forward MLP with GELU non-linearity. The normalization layer is applied before both of the two parts. Both parts employ residual connection.

3.1 Memory-friendly Generator

The task of generation poses a high standard for spatial coherency in structure, color, and texture, both globally and locally. The transformer encoders take embedding token words as inputs and calculate the interaction between each token recursively. [61, 34]. The main dilemma here is: what is the right "word" for image generation tasks? If we similarly generate an image in a pixel-by-pixel manner through stacking transformer encoders, even a low-resolution image (e.g. 32×32) can result in an excessively long sequence (1024), causing the explosive cost of self-attention (quadratic w.r.t. the sequence length) and prohibiting the scalability to higher resolutions. To avoid this daunting cost, we are inspired by a common design philosophy in CNN-based GANs, to iteratively upscale the resolution at multiple stages [25, 16]. Our strategy is hence to increase the input sequence and reduce the embedding dimension gradually .

Figure 2 (left) illustrates a memory-friendly transformer-based generator that consists of multiple stages. Each stage stacks several transformer blocks. By stages, we gradually increase the feature

map resolution until it meets the target resolution $H \times W$. Specifically, the generator takes the random noise as its input, and passes it through a multiple-layer perceptron (MLP) to a vector of length $H_0 \times W_0 \times C$. The vector is reshaped into a $H_0 \times W_0$ resolution feature map (by default we use $H_0 = W_0 = 8$), each point a C-dimensional embedding. This "feature map" is next treated as a length-64 sequence of C-dimensional tokens, combined with the learnable positional encoding.

To scale up to higher-resolution images, we insert an upsampling module after each stage, consisting of a reshaping and resolution-upscaling layer. For lower-resolution stages (resolution lower than 64×64), the upsampling module firstly reshapes the 1D sequence of token embedding back to a 2D feature map $X_i \in \mathbb{R}^{H_i \times W_i \times C}$ and then adopts the `bicubic` layer to upsample its resolution while the embedded dimension is kept unchanged, resulting in the output $X_i^{'} \in \mathbb{R}^{2H_i \times 2W_i \times C}$. After that, the 2D feature map $X_i^{'}$ is again reshaped into the 1D sequence of embedding tokens. For higher-resolution stages, we replace the `bicubic` upscaling layer with the `pixelshuffle` module, which upsamples the resolution of feature map by $2\times$ ratio and also reduces the embedding dimension to a quarter of the input. This pyramid-structure with modified upscaling layers mitigates the memory and computation explosion. We repeat multiple stages until it reaches the target resolution (H, W), and then we will project the embedding dimension to 3 and obtain the RGB image $Y \in \mathbb{R}^{H \times W \times 3}$.

3.2 Multi-scale Discriminator

Unlike the generator which synthesizes precise pixels, the discriminator is tasked to distinguish between real/fake images. This allows us to treat it as a typical classifier by simply tokenizing the input image in a coarser patch-level [34], where each patch can be regarded as a "word". However, compared to image recognition tasks where classifiers focus on the semantic differences, the discriminator executes a simpler and more detail-oriented task to distinguish between synthesized and real. Therefore, the local visual cues and artifacts will have an important effect on the discriminator. Practically, we observe that the patch splitting rule plays a crucial role, where large patch size sacrifices low-level texture details, and smaller patch size results in a longer sequence that costs more memory. The above dilemma motivates our design of multi-scale discriminator below.

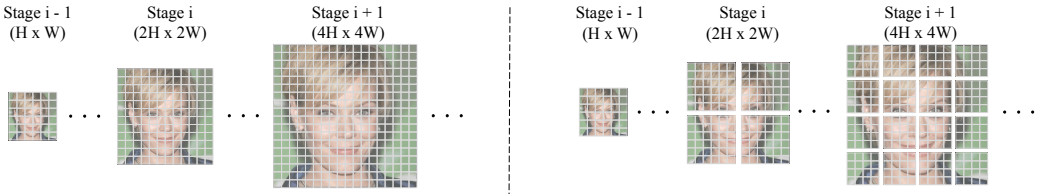
As shown in Figure 2 (right), a multi-scale discriminator is designed to take varying size of patches as inputs, at its different stages. We firstly split the input images $Y \in R^{H \times W \times 3}$ into three different sequences by choosing different patch sizes (P, $2P$, $4P$). The longest sequence $(\frac{H}{P} \times \frac{W}{P}) \times 3$ is linearly transformed to $(\frac{H}{P} \times \frac{W}{P}) \times \frac{C}{4}$ and then combined with the learnable position encoding to serve as the input of the first stage, where $\frac{C}{4}$ is the embedded dimension size. Similarly, the second and third sequences are linearly transformed to $(\frac{H}{2P} \times \frac{W}{2P}) \times \frac{C}{4}$ and $(\frac{H}{4P} \times \frac{W}{4P}) \times \frac{C}{2}$, and then separately concatenated into the second and third stages. Thus these three different sequences are able to extract both the semantic structure and texture details. Similar to the generator, we reshape the 1D-sentence to 2D feature map and adopt `Average Pooling` layer to downsample the feature map resolution, between each stage. By recursively forming the transformer blocks in each stage, we obtain a pyramid architecture where multi-scale representation is extracted. At the end of these blocks, a `[cls]` token is appended at the beginning of the 1D sequence and then taken by the classification head to output the real/fake prediction.

3.3 Grid Self-Attention: A Scalable Variant of Self-Attention for Image Generation

Self-attention allows the generator to capture the global correspondence, yet also impedes the efficiency when modeling long sequences/higher resolutions. That motivates many efficient self-attention designs in both language [62, 63] and vision tasks [64, 65]. To adapt self-attention for higher-resolution generative tasks, we propose a simple yet effective strategy, named *Grid Self-Attention*, tailored for high-resolution image generation.

As shown in Figure 3, instead of calculating the correspondence between a given token and all other tokens, the grid self-attention partitions the full-size feature map into several non-overlapped grids, and the token interactions are calculated inside each local grid. We add the grid self-attention on high-resolution stages (resolution higher than 32×32) while still keeping standard self-attention in low-resolution stages, shown as Figure 2, again so as to strategically balance local details and global awareness. The grid self-attention shows surprising effectiveness over other efficient self-attention forms [62, 65] in generative tasks, as compared later in Section 4.1.

One potential concern might arise with the boundary artifact between each grid. We observe that while the artifact indeed occurs at early training stages, it gradually vanishes given enough training



(a) Standard Self-Attention (b) Grid Self-Attention

Figure 3: Grid Self-Attention across different transformer stages. We replace Standard Self-Attention with Grid Self-Attention when the resolution is higher than 32×32 and the grid size is set to be 16×16 by default.

iterations, producing nicely coherent final results even without explicit grid overlap. We think this is owing to the larger, multi-scale receptive field of the discriminator that requires generated image fidelity in different scales. Similar observation on diminishing boundary effects was also reported in other grid-based generation works [66].

3.4 Exploring the Training Recipe

Data Augmentation. The transformer-based architectures are known to be highly data-hungry due to removing human-designed bias. Particularly in image recognition task [34], they were inferior to CNNs until much larger external data [67] was used for pre-training. To remove this roadblock, data augmentation was revealed as a blessing in [48], which showed that different types of strong data augmentation could lead us to data-efficient training for vision transformers.

We follow a similar mindset. Traditionally, training CNN-based GANs hardly refers to data augmentation. Recently, there is an interest surge in the few-shot GAN training, aiming to match state-of-the-art GAN results with orders of magnitude fewer real images [68, 69]. Contrary to this "commonsense" in CNNs, data augmentation is found to be crucial in transformer-based architectures, even with 100% real images being utilized. We show that simply using differential augmentation [68] with three basic operators $\{Translation, Cutout, Color\}$ leads to surprising performance improvement for TransGAN, while CNN-based GANs hardly benefit from it. We conduct a concrete study on the effectiveness of augmentation for both transformer and CNNs: see details in Section 4.2

Relative Position Encoding. While classical transformers [47, 34] used deterministic position encoding or learnable position encoding, the relative position encoding [70] gains increasing popularity [71, 28, 50, 72], by exploiting lags instead of absolute positions. Considering a single head of self-attention layer,

$$Attention(Q, K, V) = softmax((\frac{QK^T}{\sqrt{d_k}}V) \tag{1}$$

where Q,K,$V \in \mathbb{R}^{(H \times W) \times C}$ represent query, key, value matrices, H,W,C denotes the height, width, embedded dimension of the input feature map. The difference in coordinate between each query and key on H axis lies in the range of $[-(H-1), H-1]$, and similar for W axis. By simultaneously considering both H and W axis, the relative position can be represented by a parameterized matrix $M \in \mathbb{R}^{(2H-1) \times (2W-1)}$. Per coordinate, the relative position encoding E is taken from matrix M and added to the attention map QK^T as a bias term, shown as following,

$$Attention(Q, K, V) = softmax(((\frac{QK^T}{\sqrt{d_k}} + E)V) \tag{2}$$

Compared to its absolute counterpart, relative position encoding learns a stronger "relationship" between local contents, bringing important performance gains in large-scale cases and enjoying widespread use ever since. We also observe it to consistently improve TransGAN, especially on higher-resolution datasets. We hence apply it on top of the learnable absolute positional encoding for both the generator and discriminator.

Modified Normalization. Normalization layers are known to help stabilize the deep learning training of deep neural networks, sometimes remarkably. While both the original transformer [47] and its variants [50, 52] by default use the layer normalization, we follow previous works [73, 16] and replace it with a token-wise scaling layer to prevent the magnitudes in transformer blocks from being too high, describe as $Y = X/\sqrt{\frac{1}{C}\sum_{i=0}^{C-1}(X^i)^2 + \epsilon}$, where $\epsilon = 1e-8$ by default, X and Y denote the token before and after scaling layer, C represents the embedded dimension. Note that our modified normalization resembles local response normalization that was once used in AlexNet [73]. Unlike

Table 1: Unconditional image generation results on CIFAR-10, STl-10, and CelebA (128×128) dataset. We train the models with their official code if the results are unavailable, denoted as "*", others are all reported from references.

Methods	CIFAR-10		STL-10		CelebA
	IS↑	FID↓	IS↑	FID↓	FID↓
WGAN-GP [1]	6.49 $\pm$ 0.09	39.68	-	-	-
SN-GAN [46]	8.22 $\pm$ 0.05	-	9.16 $\pm$ 0.12	40.1	-
AutoGAN [18]	8.55 $\pm$ 0.10	12.42	9.16 $\pm$ 0.12	31.01	-
AdversarialNAS-GAN [18]	8.74 $\pm$ 0.07	10.87	9.63 $\pm$ 0.19	26.98	-
Progressive-GAN [16]	8.80 $\pm$ 0.05	15.52	-	-	7.30
COCO-GAN [66]	-	-	-	-	5.74
StyleGAN-V2 [68]	9.18	11.07	10.21* $\pm$ 0.14	20.84*	5.59*
StyleGAN-V2 + DiffAug. [68]	**9.40**	9.89	10.31* $\pm$ 0.12	19.15*	5.40*
TransGAN	9.02 $\pm$ 0.12	**9.26**	**10.43** $\pm$ 0.16	**18.28**	**5.28**

other "modern" normalization layers [74–76] that need affine parameters for both mean and variances, we find that a simple re-scaling without learnable parameters suffices to stabilize TransGAN training – in fact, it makes TransGAN train better and improves the FID.

4 Experiments

Datasets We start by evaluating our methods on three common testbeds: CIFAR-10 [77], STL-10 [78], and CelebA [79] dataset. The CIFAR-10 dataset consists of 60k 32×32 images, with 50k training and 10k testing images, respectively. We follow the standard setting to use the 50k training images without labels. For the STL-10 dataset, we use both the 5k training images and 100k unlabeled images, and all are resized to 48×48 resolution. For the CelebA dataset, we use 200k unlabeled face images (aligned and cropped version), with each image at 128×128 resolution. We further consider the CelebA-HQ and LSUN Church datasets to scale up TransGAN to higher resolution image generation tasks. We use 30k images for CelebA-HQ [16] dataset and 125k images for LSUN Church dataset [80], all at 256×256 resolution.

Implementation We follow the setting of WGAN [45], and use the WGAN-GP loss [1]. We adopt a learning rate of $1e-4$ for both generator and discriminator, an Adam optimizer with $\beta_1 = 0$ and $\beta_2 = 0.99$, exponential moving average weights for generator, and a batch size of 128 for generator and 64 for discriminator, for all experiments. We choose DiffAug. [68] as basic augmentation strategy during the training process if not specially mentioned, and apply it to our competitors for a fair comparison. Other popular augmentation strategies ([69, 10]) are not discussed here since it is beyond the scope of this work. We use common evaluation metrics Inception Score (IS) [15] and Frechet Inception Distance (FID) [81], both are measured by 50K samples with their official Tensorflow implementations [1][2]. All experiments are set with 16 V100 GPUs, using PyTorch 1.7.0. We include detailed training cost for each dataset in Appendix D. We focus on the unconditional image generation setting for simplicity.

4.1 Comparison with State-of-the-art GANs

CIFAR-10. We compare TransGAN with recently published results by unconditional CNN-based GANs on the CIFAR-10 dataset, shown in Table 1. Note that some promising conditional GANs [21, 8] are not included, due to the different settings. As shown in Table 1, TransGAN surpasses the strong model of Progressive GAN [16], and many other latest competitors such as SN-GAN [46], AutoGAN [18], and AdversarialNAS-GAN [19], in terms of inception score (IS). It is only next to the huge and heavily engineered StyleGAN-v2 [40]. Once we look at the FID results, TransGAN is even found to outperform StyleGAN-v2 [40] with both applied the same data augmentation [68].

STL-10. We then apply TransGAN on another popular benchmark STL-10, which is larger in scale (105k) and higher in resolution (48x48). We compare TransGAN with both the automatic searched and hand-crafted CNN-based GANs, shown in Table 1. Different from the results on CIFAR-10, we find that TransGAN outperforms all current CNN-based GAN models, and sets **new state-of-the-art** results in terms of both IS and FID score. This is thanks to the fact that the STL-10 dataset size is $2\times$ larger than CIFAR-10, suggesting that transformer-based architectures benefit much more notably from larger-scale data than CNNs.

[1] https://github.com/openai/improved-gan/tree/master/inception_score

[2] https://github.com/bioinf-jku/TTUR

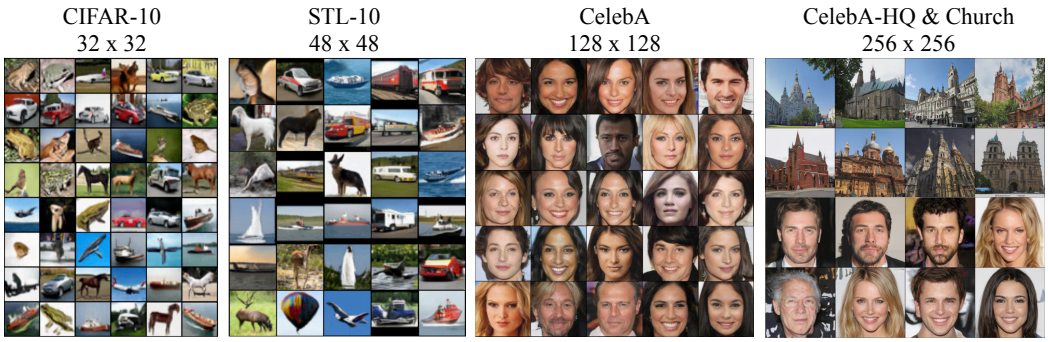


Figure 4: Visual results produced by TransGAN on different datasets, as resolution grows from 32 $\times$ 32 to 256 $\times$ 256. More visual examples are included in Appendix F.

Table 2: The effectiveness of Data Augmentation on both CNN-based GANs and TransGAN. We use the full CIFAR-10 training set and DiffAug [68].

Methods	WGAN-GP		AutoGAN		StyleGAN-V2		TransGAN	
	IS ↑	FID ↓	IS ↑	FID ↓	IS ↑	FID ↓	IS ↑	FID ↓
Original	**6.49**	39.68	8.55	**12.42**	9.18	11.07	8.36	22.53
+ DiffAug [68]	6.29	**37.14**	**8.60**	12.72	**9.40**	**9.89**	**9.02**	**9.26**

Table 3: The ablation study of proposed techniques in three common dataset CelebA(64×64), CelebA(128×128, and LSUN Church(256×256)). "OOM" represents out-of-momery issue.

Training Configuration	**CelebA** (64x64)	**CelebA** (128x128)	**LSUN Church** (256x256)
(A). Standard Self-Attention	8.92	**OOM**	**OOM**
(B). Nyström Self-Attention [62]	13.47	17.42	39.92
(C). Axis Self-Attention [65]	12.39	13.95	29.30
(D). Grid Self-Attention	9.89	10.58	20.39
+ Multi-scale Discriminator	9.28	8.03	15.29
+ Modified Normalization	7.05	7.13	13.27
+ Relative Position Encoding	6.14	6.32	11.93
(E). Converge	**5.01**	**5.28**	**8.94**

CelebA (128x128). We continue to examine another common benchmark: CelebA dataset (128×128 resolution). As shown in Table 1, TransGAN largely outperforms Progressive-GAN [16] and COCO-GAN [66], and is slightly better than the strongest competitor StyleGAN-v2 [40], by reaching a FID score of 5.28. Visual examples generated on CIFAR-10, STL-10, and CelebA (128×128) are shown in Figure 4, from which we observe pleasing visual details and diversity.

4.2 Scaling Up to Higher-Resolution

We further scale up TransGAN to higher-resolution (256×256) generation, including on CelebA-HQ [16] and LSUN Church [80]. These high-resolution datasets are significantly more challenging due to their much richer and detailed low-level texture as well as the global composition. Thanks to the proposed multi-scale discriminator, TransGAN produces pleasing visual results, reaching competitive quantitative results with 10.28 FID on CelebA-HQ 256×256 and 8.94 FID on LSUN Church dataset, respectively. As shown in Figure 4, diverse examples with rich textures details are produced. We discuss the memory cost reduction brought by the Grid Self-Attention in Appendix E.

4.3 Data Augmentation is Crucial for TransGAN

We study the effectiveness of data augmentation for both CNN-based GANs and Our TransGAN. We apply the differentiable augmentation [68] to all these methods. As shown in Table 2, for three CNN-based GANs, the performance gains of data augmentation seems to diminish in the full-data regime. Only the largest model, StyleGAN-V2, is improved on both IS and FID. In sharp contrast, TransGAN sees a shockingly large margin of improvement: IS improving from 8.36 to 9.02 and FID improving from 22.53 to 9.26. This phenomenon suggests that CIFAR-10 is still "small-scale " when

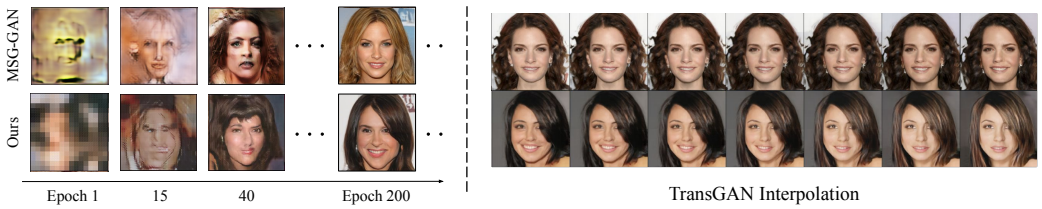


Figure 5: Left: training dynamic with training epochs for both TransGAN and MSG-GAN on CelebA-HQ (256×256). Right: Interpolation on latent space produced by TransGAN

fitting transformers; it re-confirms our assumption that transformer-based architectures are much more data-hungry than CNNs, and that can be helped by stronger data augmentation.

4.4 Ablation Study

To further evaluate the proposed grid self-attention, multi-scale discriminator, and unique training recipe, we conduct the ablation study by separately adding these techniques to the baseline method and report their FID score on different datasets. Due to the fact that most of our contributions are tailored for the challenges brought by higher-resolution tasks, we choose CelebA and LSUN Church as the main testbeds, with details shown in Table 3. We start by constructing our memory-friendly with vanilla discriminator as our baseline method (A), both applied with standard self-attention. The baseline method achieves relatively good results with 8.92 FID on CelebA (64×64) dataset, however, it fail on higher-resolution tasks due to the memory explosion issue brought by self-attention. This motivates us to evaluate two efficient form of self-attention, (B) `Nyström Self-Attention` [62] and (C) `Axis Self-Attention` [65]

By replacing all self-attention layers in high-resolution stages (feature map resolution higher than 32×32) with these efficient variants, both two methods (B)(C) are able to produce reasonable results. However, they still show to be inferior to standard self-attention, even on the 64×64 resolution dataset. By adopting our proposed `Grid Self-Attention` (D), we observe a significant improvement on both three datasets, reaching 9.89, 10.58, 20.39 FID on CelebA 64×64, 128×128 and LSUN Church 256×256, respectively. Based on the configuration (D), we continue to add the proposed techniques, including the multi-scale discriminator, modified normalization, and relative position encoding. All these three techniques significantly improve the performance of TransGAN on three datasets. At the end, we train our final configuration (E) until it converges, resulting in the best FID on CelebA 64×64 (**5.01**), CelebA 128×128 (**5.28**), and LSUN Church 256×256 (**8.94**).

4.5 Understanding Transformer-based Generative Model

We dive deep into our transformer-based GAN by conducting interpolation on latent space and comparing its behavior with CNN-based GAN, through visualizing their training dynamics. We choose MSG-GAN [82] for comparison since it extracts multi-scale representation as well. As shown in Figure 5, the CNN-based GAN quickly extracts face representation in the early stage of training process while transformer only produces rough pixels with no meaningful global shape due to missing any inductive bias. However, given enough training iterations, TransGAN gradually learns informative position representation and is able to produce impressive visual examples at convergence. Meanwhile, the boundary artifact also vanishes at the end. For the latent space interpolation, TransGAN continues to show encouraging results where smooth interpolation are maintained on both local and global levels. More high-resolution visual examples will be presented in Appendix F.

5 Conclusions, Limitation, and Discussions of Broad Impact

In this work, we provide the first pilot study of building GAN with pure transformers. We have carefully crafted the architectures and thoughtfully designed training techniques. As a result, the proposed TransGAN has achieved state-of-the-art performance across multiple popular datasets, and easily scales up to higher-resolution generative tasks. Although TransGAN provides an encouraging starting point, there is still a large room to explore further, such as going towards extremely high resolution generation tasks (e.g., 1024×1024), which would be our future directions.

Broader Impact. The proposed generative model can serve as a data engine to address data collection issues. More importantly, using synthesized image examples helps avoid privacy concerns. By making generative model accessible to more people, TransGAN provides a convenient way to help everybody

make artistic creations. However, the abuse of generative models may also create more fake news, which should be more effectively restricted in the future.

6 Acknowledgements

We would like to express our deepest gratitude to the MIT-IBM Watson AI Lab, in particular John Cohn for generously providing us with the computing resources necessary to conduct this research.

References

[1] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

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

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[5] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in neural information processing systems*, pages 465–476, 2017.

[6] Shuai Yang, Zhangyang Wang, Zhaowen Wang, Ning Xu, Jiaying Liu, and Zongming Guo. Controllable artistic text style transfer via shape-matching gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4442–4451, 2019.

[7] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. Enlightengan: Deep light enhancement without paired supervision. *IEEE Transactions on Image Processing*, 30:2340–2349, 2021.

[8] Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pages 3581–3590. PMLR, 2019.

[9] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in neural information processing systems*, pages 2018–2028, 2017.

[10] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. *arXiv preprint arXiv:1910.12027*, 2019.

[11] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

[12] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[13] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[14] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in neural information processing systems*, pages 2203–2213, 2017.

[15] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[16] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[17] Mario Lucic, Karol Kurach, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Are gans created equal? a large-scale study. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 698–707, 2018.

[18] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3224–3234, 2019.

[19] Chen Gao, Yunpeng Chen, Si Liu, Zhenxiong Tan, and Shuicheng Yan. Adversarialnas: Adversarial neural architecture search for gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5680–5689, 2020.

[20] Yuan Tian, Qin Wang, Zhiwu Huang, Wen Li, Dengxin Dai, Minghao Yang, Jun Wang, and Olga Fink. Off-policy reinforcement learning for efficient and effective gan architecture search. In *European Conference on Computer Vision*, pages 175–192. Springer, 2020.

[21] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[22] 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*, pages 4401–4410, 2019.

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

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

[25] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *arXiv preprint arXiv:1506.05751*, 2015.

[26] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[27] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[28] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019.

[29] Yu-Syuan Xu, Shou-Yao Roy Tseng, Yu Tseng, Hsien-Kai Kuo, and Yi-Min Tsai. Unified dynamic convolutional network for super-resolution with variational degradations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12496–12505, 2020.

[30] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[31] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2502–2510, 2018.

[32] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

[33] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *ECCV*. Springer, 2020.

[34] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[35] Drew A Hudson and C Lawrence Zitnick. Generative adversarial transformers. *arXiv preprint arXiv:2103.01209*, 2021.

[36] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020.

[37] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020.

[38] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *arXiv preprint arXiv:2001.06937*, 2020.

[39] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8207–8216, 2020.

[40] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[41] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[42] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8878–8887, 2019.

[43] Xi Ouyang, Yu Cheng, Yifan Jiang, Chun-Liang Li, and Pan Zhou. Pedestrian-synthesis-gan: Generating pedestrian data in real scene and beyond. *arXiv preprint arXiv:1804.02047*, 2018.

[44] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

[45] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[46] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[49] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021.

[50] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[51] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*, 2021.

[52] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

[53] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *arXiv preprint arXiv:2103.12424*, 2021.

[54] Yutong Xie, Jianpeng Zhang, Chunhua Shen, and Yong Xia. Cotr: Efficiently bridging cnn and transformer for 3d medical image segmentation. *arXiv preprint arXiv:2103.03024*, 2021.

[55] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020.

[56] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020.

[57] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[58] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2020.

[59] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[60] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[62] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nystr\" omformer: A nystr\" om-based algorithm for approximating self-attention. *arXiv preprint arXiv:2102.03902*, 2021.

[63] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[64] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[65] Manoj Kumar, Dirk Weissenborn, and Nal Kalchbrenner. Colorization transformer. *arXiv preprint arXiv:2102.04432*, 2021.

[66] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. Coco-gan: Generation by parts via conditional coordinating. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4512–4521, 2019.

[67] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

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

[69] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020.

[70] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[71] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[72] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, AM Dai, MD Hoffman, and D Eck. Music transformer: Generating music with long-term structure (2018). *arXiv preprint arXiv:1809.04281*, 2018.

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[74] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

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

[77] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[78] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[79] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[80] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, 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.

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

[82] Animesh Karnewar and Oliver Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7799–7808, 2020.

A Zoomed-in High-resolution Examples

The high-resolution (especially 256×256) visual examples are resized to fit the layout of our main manuscript. Here we provide the zoom-in version of those examples on 256×256 dataset to show their texture details. As shown in Figure. 6,7, the visual examples produced by TransGAN show high fidelity with rich texture details. More visual results are shown in Appendix F.

B Implementation of Data Augmentation

We mainly follow the way of differentiable augmentation [68] to apply the data augmentation on our GAN training framework. Specifically, we conduct $\{Translation, Cutout, Color\}$ augmentation for TransGAN with probability p, while p is empirically set to be $\{1.0, 0.3, 1.0\}$. However, we find that $Translation$ augmentation will hurt the performance of CNN-based GAN when 100% data is utilized. Therefore, we remove it and only conduct $\{Cutout, Color\}$ augmentation for AutoGAN [18]. We also evaluate the effectiveness of stronger augmentation on high-resolution generative tasks (E.g. 256×256), including `random-cropping`, `random hue adjustment`, and `image filtering`. We show that it can further improve the FID score on CelebA-HQ from 10.28 to 9.60. Moreover, we find `image filtering` helps remove the boundary artifacts in a very early stage of training process, while it takes longer training iterations to remove it in the original setting.

C Detailed Architecture Configurations

We present the specific architecture configurations of TransGAN on different datasets, shown in Table 4, 5, 6, 7. For the generator architectures, the "Block" represents the basic Transformer Block constructed by self-atention, Normalization, and Feed-forward MLP. "Grid Block" denotes the Transformer Block where the standard self-attention is replaced by the propose `Grid Self-Attention`, with grid size equals to 16. `Upsampling` layer represents `Bicubic Upsampling` by default. The "input_shape" and "output_shape" denotes the shape of input feature map and output feature map, respectively. For the discriminator architectures, we use "Layer Flatten" to represent the process of patch splitting and linear transformation. In each stage, the output feature map is concatenated with another different sequence, as described in Sec. 3.2. In the final stage, we add another `CLS` token and use a Transformer Block to build correspondence between `CLS` token and extracted representation. In the end, only the `CLS` token is taken by the Classification Head for predicting real/fake. For low-resolution generative tasks (e.g., CIFAR-10 and STL-10), we only split the input images into two different sequences rather than three and only two stages are built as well.

D Training Cost

We include the training cost of TransGAN on different datasets, with resolutions across from 32×32 to 256×256, shown in Table 8. The largest experiment costs around 4 days with 16 V100 GPUs.

Figure 6: High-resolution representative visual examples.

E Memory Cost Comparison

We compare the GPU memory cost between standard self-attention and grid self-attention. Our testbed is set on Nvidia V100 GPU with batch size set to 1, using Pytorch V1.7 environment. We evaluate the inference cost of these two architectures, without calculating the gradient. Since the original self-attention will cause out-of-memory issue even when batch size is set to 1, we reduce the model size on (256×256) resolution tasks to make it fit GPU memory, and apply the same strategy on 128×128 and 64×64 architectures as well. When evaluating the grid self-attention, we do not reduce the model size and only modify the standard self-attention on the specific stages where the resolution is larger than 32×32, and replace it with the proposed Grid Self-Attention. As shown in in Figure 8, even the model size of the one that represents the standard self-attention is reduced, it still costs significantly larger GPU memory than the proposed Grid Self-Attention does.

Figure 7: High-resolution representative visual examples.

Table 4: Architecture configuration of TransGAN on CIFAR-10 dataset.

Generator			
Stage	Layer	Input Shape	Output Shape
-	MLP	512	$(8 \times 8) \times 1024$
1	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
2	PixelShuffle	$(8 \times 8) \times 1024$	$(16 \times 16) \times 256$
	Block	$(16 \times 16) \times 256$	$(16 \times 16) \times 256$
	Block	$(16 \times 16) \times 256$	$(16 \times 16) \times 256$
	Block	$(16 \times 16) \times 256$	$(16 \times 16) \times 256$
	Block	$(16 \times 16) \times 256$	$(16 \times 16) \times 256$
3	PixelShuffle	$(16 \times 16) \times 256$	$(32 \times 32) \times 64$
	Block	$(32 \times 32) \times 64$	$(32 \times 32) \times 64$
	Block	$(32 \times 32) \times 64$	$(32 \times 32) \times 64$
-	Linear Layer	$(32 \times 32) \times 64$	$32 \times 32 \times 3$

Discriminator			
Stage	Layer	Input Shape	Out Shape
-	Linear Flatten	$32 \times 32 \times 3$	$(16 \times 16) \times 192$
1	Block	$(16 \times 16) \times 192$	$(16 \times 16) \times 192$
	Block	$(16 \times 16) \times 192$	$(16 \times 16) \times 192$
	Block	$(16 \times 16) \times 192$	$(16 \times 16) \times 192$
	AvgPooling	$(16 \times 16) \times 192$	$(8 \times 8) \times 192$
	Concatenate	$(8 \times 8) \times 192$	$(8 \times 8) \times 384$
2	Block	$(8 \times 8) \times 384$	$(8 \times 8) \times 384$
	Block	$(8 \times 8) \times 384$	$(8 \times 8) \times 384$
	Block	$(8 \times 8) \times 384$	$(8 \times 8) \times 384$
-	Add CLS Token	$(8 \times 8) \times 384$	$(8 \times 8 + 1) \times 384$
	Block	$(8 \times 8 + 1) \times 384$	$(8 \times 8 + 1) \times 384$
	CLS Head	1×384	1

F Visual Examples

We include more high-resolution visual examples on Figure 9,10. The visual examples produced by TransGAN show impressive details and diversity.

Table 5: Architecture configuration of TransGAN on STL-10 dataset.

Generator			
Stage	Layer	Input Shape	Output Shape
-	MLP	512	$(12 \times 12) \times 1024$
1	Block	$(12 \times 12) \times 1024$	$(12 \times 12) \times 1024$
	Block	$(12 \times 12) \times 1024$	$(12 \times 12) \times 1024$
	Block	$(12 \times 12) \times 1024$	$(12 \times 12) \times 1024$
	Block	$(12 \times 12) \times 1024$	$(12 \times 12) \times 1024$
	Block	$(12 \times 12) \times 1024$	$(12 \times 12) \times 1024$
2	PixelShuffle	$(12 \times 12) \times 1024$	$(24 \times 24) \times 256$
	Block	$(24 \times 24) \times 256$	$(24 \times 24) \times 256$
	Block	$(24 \times 24) \times 256$	$(24 \times 24) \times 256$
	Block	$(24 \times 24) \times 256$	$(24 \times 24) \times 256$
	Block	$(24 \times 24) \times 256$	$(24 \times 24) \times 256$
3	PixelShuffle	$(24 \times 24) \times 256$	$(48 \times 48) \times 64$
	Block	$(48 \times 48) \times 64$	$(48 \times 48) \times 64$
	Block	$(48 \times 48) \times 64$	$(48 \times 48) \times 64$
-	Linear Layer	$(48 \times 48) \times 64$	$48 \times 48 \times 3$

Discriminator			
Stage	Layer	Input Shape	Out Shape
-	Linear Flatten	$48 \times 48 \times 3$	$(16 \times 16) \times 192$
1	Block	$(24 \times 24) \times 192$	$(24 \times 24) \times 192$
	Block	$(24 \times 24) \times 192$	$(24 \times 24) \times 192$
	Block	$(24 \times 24) \times 192$	$(24 \times 24) \times 192$
	AvgPooling	$(24 \times 24) \times 192$	$(12 \times 12) \times 192$
	Concatenate	$(12 \times 12) \times 192$	$(12 \times 12) \times 384$
2	Block	$(12 \times 12) \times 384$	$(12 \times 12) \times 384$
	Block	$(12 \times 12) \times 384$	$(12 \times 12) \times 384$
	Block	$(12 \times 12) \times 384$	$(12 \times 12) \times 384$
-	Add CLS Token	$(12 \times 12) \times 384$	$(12 \times 12 + 1) \times 384$
	Block	$(12 \times 12 + 1) \times 384$	$(12 \times 12 + 1) \times 384$
	CLS Head	1×384	1

Table 6: Architecture configuration of TransGAN on CelebA (128×128) dataset.

Generator			
Stage	Layer	Input Shape	Output Shape
-	MLP	512	$(8 \times 8) \times 1024$
1	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
2	Upsampling	$(8 \times 8) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
3	PixelShuffle	$(16 \times 16) \times 1024$	$(32 \times 32) \times 256$
	Block	$(32 \times 32) \times 256$	$(32 \times 32) \times 256$
	Block	$(32 \times 32) \times 256$	$(32 \times 32) \times 256$
	Block	$(32 \times 32) \times 256$	$(32 \times 32) \times 256$
	Block	$(32 \times 32) \times 256$	$(32 \times 32) \times 256$
4	PixelShuffle	$(32 \times 32) \times 256$	$(64 \times 64) \times 64$
	Grid Block	$(64 \times 64) \times 64$	$(64 \times 64) \times 64$
	Grid Block	$(64 \times 64) \times 64$	$(64 \times 64) \times 64$
	Grid Block	$(64 \times 64) \times 64$	$(64 \times 64) \times 64$
	Grid Block	$(64 \times 64) \times 64$	$(64 \times 64) \times 64$
5	PixelShuffle	$(64 \times 64) \times 64$	$(128 \times 128) \times 16$
	Grid Block	$(128 \times 128) \times 16$	$(128 \times 128) \times 16$
	Grid Block	$(128 \times 128) \times 16$	$(128 \times 128) \times 16$
	Grid Block	$(128 \times 128) \times 16$	$(128 \times 128) \times 16$
	Grid Block	$(128 \times 128) \times 16$	$(128 \times 128) \times 16$
-	Linear Layer	$(128 \times 128) \times 16$	$128 \times 128 \times 3$

Discriminator			
Stage	Layer	Input Shape	Out Shape
-	Linear Flatten	$128 \times 128 \times 3$	$(32 \times 32) \times 96$
1	Block	$(32 \times 32) \times 96$	$(32 \times 32) \times 96$
	Block	$(32 \times 32) \times 96$	$(32 \times 32) \times 96$
	Block	$(32 \times 32) \times 96$	$(32 \times 32) \times 96$
	AvgPooling	$(32 \times 32) \times 96$	$(16 \times 16) \times 96$
	Concatenate	$(16 \times 16) \times 96$	$(16 \times 16) \times 192$
2	Block	$(16 \times 16) \times 192$	$(16 \times 16) \times 192$
	Block	$(16 \times 16) \times 192$	$(16 \times 16) \times 192$
	Block	$(16 \times 16) \times 192$	$(16 \times 16) \times 192$
	AvgPooling	$(16 \times 16) \times 192$	$(8 \times 8) \times 192$
	Concatenate	$(8 \times 8) \times 192$	$(8 \times 8) \times 384$
3	Block	$(8 \times 8) \times 192$	$(8 \times 8) \times 384$
	Block	$(8 \times 8) \times 384$	$(8 \times 8) \times 384$
	Block	$(8 \times 8) \times 384$	$(8 \times 8) \times 384$
-	Add CLS Token	$(8 \times 8) \times 384$	$(8 \times 8 + 1) \times 384$
	Block	$(8 \times 8 + 1) \times 384$	$(8 \times 8 + 1) \times 384$
	CLS Head	1×384	1

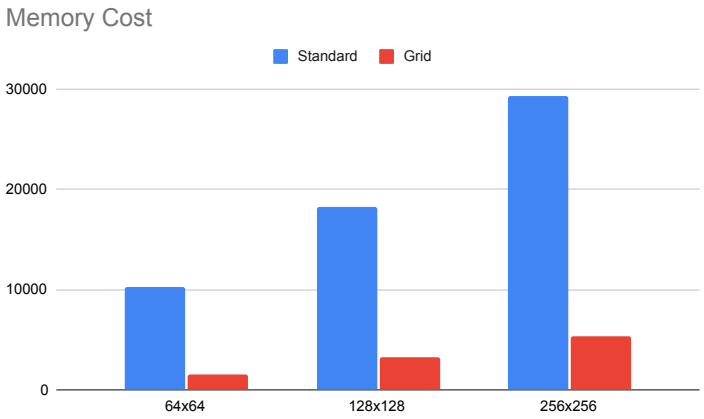


Figure 8: Memory cost comparison between standard self-attention and grid self-attention

Table 7: Architecture configuration of TransGAN on CelebA (256×256) and LSUN Church (256×256) dataset.

Generator			
Stage	Layer	Input Shape	Output Shape
-	MLP	512	$(8 \times 8) \times 1024$
1	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
	Block	$(8 \times 8) \times 1024$	$(8 \times 8) \times 1024$
2	Upsampling	$(8 \times 8) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
	Block	$(16 \times 16) \times 1024$	$(16 \times 16) \times 1024$
3	Upsampling	$(16 \times 16) \times 1024$	$(32 \times 32) \times 1024$
	Block	$(32 \times 32) \times 1024$	$(32 \times 32) \times 1024$
	Block	$(32 \times 32) \times 1024$	$(32 \times 32) \times 1024$
	Block	$(32 \times 32) \times 1024$	$(32 \times 32) \times 1024$
	Block	$(32 \times 32) \times 1024$	$(32 \times 32) \times 1024$
4	PixelShuffle	$(32 \times 32) \times 1024$	$(64 \times 64) \times 256$
	Grid Block	$(64 \times 64) \times 256$	$(64 \times 64) \times 256$
	Grid Block	$(64 \times 64) \times 256$	$(64 \times 64) \times 256$
	Grid Block	$(64 \times 64) \times 256$	$(64 \times 64) \times 256$
	Grid Block	$(64 \times 64) \times 256$	$(64 \times 64) \times 256$
5	PixelShuffle	$(64 \times 64) \times 256$	$(128 \times 128) \times 64$
	Grid Block	$(128 \times 128) \times 64$	$(128 \times 128) \times 64$
	Grid Block	$(128 \times 128) \times 64$	$(128 \times 128) \times 64$
	Grid Block	$(128 \times 128) \times 64$	$(128 \times 128) \times 64$
	Grid Block	$(128 \times 128) \times 64$	$(128 \times 128) \times 64$
6	PixelShuffle	$(128 \times 128) \times 64$	$(256 \times 256) \times 16$
	Grid Block	$(256 \times 256) \times 16$	$(256 \times 256) \times 16$
	Grid Block	$(256 \times 256) \times 16$	$(256 \times 256) \times 16$
	Grid Block	$(256 \times 256) \times 16$	$(256 \times 256) \times 16$
	Grid Block	$(256 \times 256) \times 16$	$(256 \times 256) \times 16$
-	Linear Layer	$(256 \times 256) \times 16$	$256 \times 256 \times 3$

Discriminator			
Stage	Layer	Input Shape	Out Shape
-	Linear Flatten	$256 \times 256 \times 3$	$(64 \times 64) \times 96$
1	Block	$(64 \times 64) \times 96$	$(64 \times 64) \times 96$
	Block	$(64 \times 64) \times 96$	$(64 \times 64) \times 96$
	Grid Block	$(64 \times 64) \times 96$	$(64 \times 64) \times 96$
	AvgPooling	$(64 \times 64) \times 96$	$(32 \times 32) \times 96$
	Concatenate	$(32 \times 32) \times 96$	$(32 \times 32) \times 192$
2	Block	$(32 \times 32) \times 192$	$(32 \times 32) \times 192$
	Block	$(32 \times 32) \times 192$	$(32 \times 32) \times 192$
	Block	$(32 \times 32) \times 192$	$(32 \times 32) \times 192$
	AvgPooling	$(32 \times 32) \times 192$	$(16 \times 16) \times 192$
	Concatenate	$(16 \times 16) \times 192$	$(16 \times 16) \times 384$
3	Block	$(16 \times 16) \times 192$	$(16 \times 16) \times 384$
	Block	$(16 \times 16) \times 384$	$(16 \times 16) \times 384$
	Block	$(16 \times 16) \times 384$	$(16 \times 16) \times 384$
-	Add CLS Token	$(16 \times 16) \times 384$	$(16 \times 16 + 1) \times 384$
	Block	$(16 \times 16 + 1) \times 384$	$(16 \times 16 + 1) \times 384$
	CLS Head	1×384	1

Table 8: Training Configuration

Dataset	Size	Resolution	GPUs	Epochs	Time
CIFAR-10	50k	32×32	2	500	2.6 days
STL-10	105k	48×48	4	200	2.0 days
CelebA	200k	64×64	8	250	2.4 days
CelebA	200k	128×128	16	250	2.5 days
CelebA-HQ	30k	256×256	16	300	3.1 days
LSUN Church	125k	256×256	16	120	4.3 days

Figure 9: Latent Space Interpolation on CelebA (256×256) dataset.

Figure 10: High-resolution representative visual examples on CelebA (256×256) dataset.