

StudioGAN: A Taxonomy and Benchmark of GANs for Image Synthesis

Minguk Kang, Joonghyuk Shin, Jaesik Park,

Abstract—Generative Adversarial Network (GAN) is one of the state-of-the-art generative models for realistic image synthesis. While training and evaluating GAN becomes increasingly important, the current GAN research ecosystem does not provide reliable benchmarks for which the evaluation is conducted consistently and fairly. Furthermore, because there are few validated GAN implementations, researchers devote considerable time to reproducing baselines. We study the taxonomy of GAN approaches and present a new open-source library named StudioGAN. StudioGAN supports 7 GAN architectures, 9 conditioning methods, 4 adversarial losses, 12 regularization modules, 3 differentiable augmentations, 7 evaluation metrics, and 5 evaluation backbones. With our training and evaluation protocol, we present a large-scale benchmark using various datasets (CIFAR10, ImageNet, AFHQv2, FFHQ, and Baby/Papa/Grampa-ImageNet) and 3 different evaluation backbones (InceptionV3, SwAV, and Swin Transformer). Unlike other benchmarks used in the GAN community, we train representative GANs, including BigGAN and StyleGAN series in a unified training pipeline and quantify generation performance with 7 evaluation metrics. The benchmark evaluates other cutting-edge generative models (e.g., StyleGAN-XL, ADM, MaskGIT, and RQ-Transformer). StudioGAN provides GAN implementations, training, and evaluation scripts with the pre-trained weights. StudioGAN is available at <https://github.com/POSTECH-CVLab/PyTorch-StudioGAN>.

Index Terms—Realistic image synthesis, Taxonomy of generative adversarial networks, and Benchmark of generative models

1 INTRODUCTION

GENERATIVE Adversarial Network (GAN) [1] is a well-known paradigm for approximating a real data distribution through an adversarial process. A GAN pipeline comprises a generator and a discriminator network. The discriminator tries to classify whether given samples are from the generator or the real data. On the other hand, the generator strives to generate realistic examples to induce the discriminator to misjudge the generated samples as real. By repeating the adversarial process, it is demonstrated that the generator can approximate the real data distribution [1].

Since Radford *et al.* [2] showed that deep convolutional GAN (DCGAN) can generate high-quality natural images, GAN has been actively adopted for a variety of computer vision applications, including natural image synthesis [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], image-to-image translation [12], [13], [14], [15], [16], [17], super-resolution imaging [18], [19], [20], [21], and generative neural radiance field [22], [23], [24], [25], [26]. Along with the increasing demand for developing a high-performance generative model, evaluating the quality of generated images has become an imperative procedure for GAN development. Inception score (IS) [27], and Fréchet Inception Distance (FID) [28] are the representative metrics for the generative model evaluation, and follow-up studies adopt the two metrics for the comparison.

While researchers put much effort into properly evaluating GAN, there are still inappropriate training configurations, and evaluation procedures are inconsistent between

proposed approaches. For instance, some software libraries show unexpected behavior in data processing procedures. A quantization error occurs when converting training images to an hdf5 format file for fast data I/O. Similarly, buggy image resizing libraries can cause aliasing artifacts in images, affecting the quality of generated images [9], [29]. The lack of elaborately designed training and evaluation protocols followed by reliable benchmark, impede the community's advancement in developing better generative models.

Generative model evaluation is mainly concerned with IS [27] and FID [28] where a pre-trained InceptionV3 [30] is used to extract the features of images. As stated by Kynkänniemi *et al.* [31], FID can be hacked if information leakage from the InceptionV3 network takes place. This implies the importance of comprehensive benchmark with various metrics and evaluation backbones, such as IS, FID, Precision & Recall [32], and Density & Coverage [33], evaluated using different backbones, in order to thoroughly examine various aspects of models.

In this work, we introduce a new software library, named **StudioGAN** which has implementations of 30 representative GANs, covering from DCGAN [2] to StyleGAN3 [9] reproducing reported results in most cases. StudioGAN provides 7 GAN architectures, 9 conditioning methods, 4 adversarial losses, 12 regularization methods, and 7 evaluation metrics. StudioGAN allows training and evaluating representative GANs in the same environment, indeed with proper training and evaluation protocols for a fair comparison. Moreover, StudioGAN allows the change of evaluation backbone from InceptionV3 to modern networks, including SwAV [34], [35], DINO [36], and Swin-T [37]. As a result, we present an unparalleled amount of benchmark results instead of referring to the results from existing papers.

Note that there are several papers [38], [39], [40], [41],

• Minguk Kang is with POSTECH, Pohang, Republic of Korea, 37673. Joonghyuk Shin, and Jaesik Park are with Seoul National University, Seoul, Republic of Korea, 08826

[42] that provide benchmark tables of generative models. In these papers, authors contain the evaluation results from previous papers without being fully aware that the original papers often utilize different training and evaluation protocols. Correspondingly, these practices still result in unfairness due to the disparities in data preprocessing, the employed machine learning library, and some unintentional mistakes by researchers. Recent research by Parmar *et al.* [29] established protocols to train and evaluate GANs in a more controlled setup. While their primary goal is to establish a protocol for fairer comparison between generative models, the benchmark they present is limited to models with open-source implementations and includes a re-evaluation of pre-trained models. This motivates us to establish an extensive benchmark that is controlled from the scratch-training to fair and clear evaluation. By providing identical training and evaluating conditions, we aim to provide a playground where researchers can easily implement their new ideas, followed by an extensive, fairly computed benchmark.

While the primary focus of our paper is to provide open-source implementations of GANs with an extensive benchmark, we also include a comparison between GANs and other generative models, such as diffusion models [43], [44], [45], [46], [47] and auto-regressive models [40], [41], [48] in Section 8. GANs noticeably differ from diffusion and auto-regressive models regarding the number of feed-forward processes for generating a single sample. Therefore, we present the inference speed and the number of model parameters to help analyze the strengths and weaknesses of each model. For diffusion models or auto-regressive models, we use official implementations and apply the same evaluation protocol used for GAN evaluation.

In our extensive evaluation of GANs, we observe interesting findings on BigGAN, StyleGAN2, and StyleGAN3 characteristics. For example, we experimentally identify that StyleGAN2 tends to generate more diverse but lower quality images than BigGAN. However, we also witness that training StyleGAN2 and StyleGAN3 is more challenging than BigGAN if the targeted distribution has large inter-class variations. Furthermore, we discuss the potential hazards that can arise during a comparison with other models and add the future research direction at the end of the paper. To the best of our knowledge, this is the first attempt to provide a fairly established benchmark for generative modeling on such a substantial scale.

In summary, this paper has the following contributions.

- We present a taxonomy of GAN approaches based on five criteria (Sec. 3).
- We pinpoint inappropriate practices that can cause poor generation results, reproducibility issues, and unfair evaluation (Sec. 4).
- We introduce StudioGAN, a software library that consists of 7 GAN architectures, 9 conditioning methods, 4 adversarial losses, 12 regularization modules, 3 differentiable augmentations, 7 evaluation metrics, and 5 evaluation backbones (Sec. 6). Through carefully assembling provided modules, researchers can design diverse and reproducible GANs, including 30 existing GANs.
- We provide large-scale evaluation benchmark for generative model development on standard and novel

datasets (Sec. 7). Some interesting findings from modern generative models are presented (Sec. A4 and 8).

2 PRELIMINARY

Generative Adversarial Network. Goodfellow *et al.* [1] proposed a framework for generative modeling based on a two-player minimax game. Generative Adversarial Network (GAN) aims to estimate the distribution of an arbitrary data in high-dimensional space and consists of a generator and a discriminator network adversarial to each other.

Specifically, the *generator* $G : \mathcal{Z} \rightarrow \mathcal{X}$ strives to map a latent variable $\mathbf{z} \sim p(\mathbf{z})$ into a sample $G(\mathbf{z})$ as if the sample is drawn from the real data distribution $p(\mathbf{x})$. The *discriminator* $D : \mathcal{X} \rightarrow [0, 1]$, on the other hand, tries to discriminate whether the given sample comes from the real data distribution or the generator, becoming the adversary to the generator. Mathematically, the generative and discriminative processes can be expressed as follows:

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]. \quad (1)$$

The adversarial training between the two networks converges to Nash equilibrium under the two assumptions: (1) the two models have enough capacity, and (2) the discriminator is allowed to reach its optimum against the generator at each step. Under these conditions, implicit distribution of the generator $p_{G(\mathbf{z})}(\mathbf{x})$ converges to the true data distribution $p(\mathbf{x})$, and generator becomes an efficient sampler for the true data distribution.

Evaluating GANs. To quantify the generated image's quality without laborious human intervention, Inception Score (**IS**) [27] and Fréchet Inception Distance (**FID**) [28] were introduced and have been broadly adopted. Those metrics are computed on feature space, so all images involved in computing the metrics are mapped to the feature space using a pre-trained InceptionV3 [30] network. IS can be written as follows.

$$\text{IS}(\mathbf{X}_t) = \text{Score}(\mathbf{X}_t, F = \text{InceptionV3}) \\ = \exp \left(\frac{1}{M} \sum_{i=1}^M D_{\text{KL}} \left(p(y|\mathbf{x}_i) \mid \hat{p}(y) \right) \right), \quad (2)$$

where $\mathbf{X}_t = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ is the image samples we target to evaluate, the posterior $p(y|\mathbf{x}_i)$ is estimated by F (InceptionV3), D_{KL} is Kullback–Leibler divergence, and $\hat{p}(y)$ is an empirical class distribution: $\frac{1}{M} \sum_{i=1}^M p(y|\mathbf{x}_i)$. FID is defined as follows.

$$\text{FID}(\mathbf{X}_s, \mathbf{X}_t) = \text{FD}(\mathbf{X}_s, \mathbf{X}_t, F = \text{InceptionV3}) \\ = \|\mu_s - \mu_t\|_2^2 + \text{Tr} \left(\Sigma_s + \Sigma_t - 2(\Sigma_s \Sigma_t)^{\frac{1}{2}} \right), \quad (3)$$

where μ and Σ are the mean vector and covariance matrix of the features from F , and the subscripts s and t indicate the source and target, respectively.

3 TAXONOMY OF GAN APPROACHES

Since the invention of the GAN framework, enormous efforts have been made to utilize GANs for realistic data generation. We present a GAN taxonomy based on five

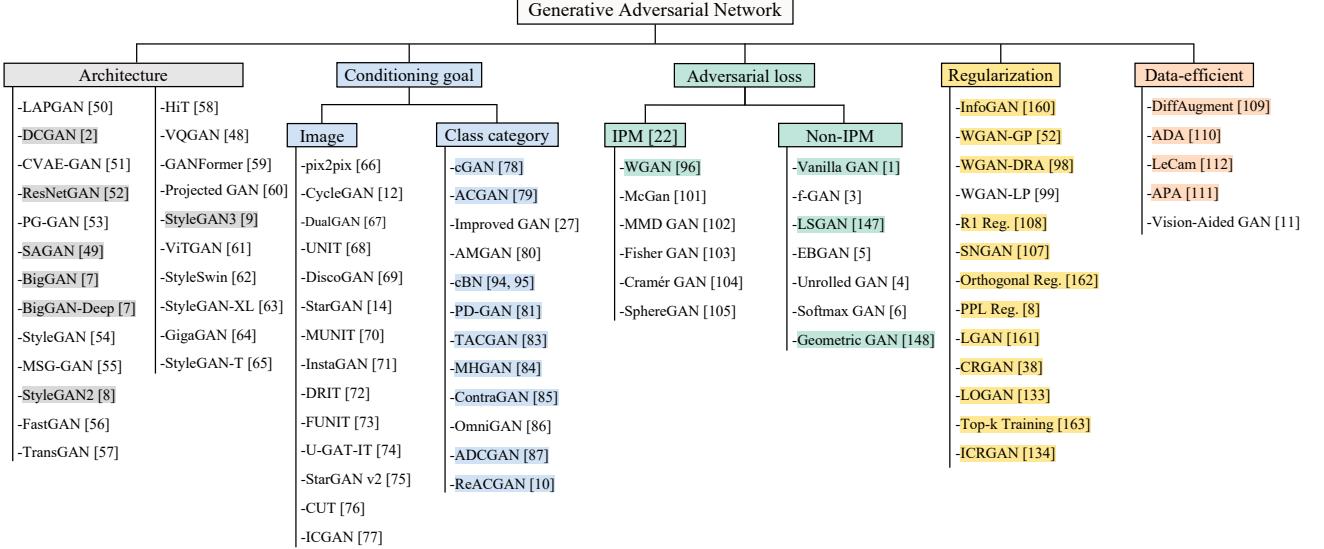


Figure 1. Taxonomy of GAN. The taxonomy comprises five divisions: architecture, conditioning goal, adversarial loss, regularization, and data-efficient training techniques. The colored elements indicate the models implemented in our proposed StudioGAN platform. StudioGAN provides 7 GAN architectures, 9 conditioning methods, 4 adversarial losses, 12 regularizations, and 5 data-efficient training methods. We will evaluate multiple GANs using 7 evaluation metrics (IS [27], FID [28], Precision [32], Recall [32], Density [33], Coverage [33], and Intra-class FID [49]) in section 7.

divisions: architecture, conditioning goal, adversarial loss, regularization, and data-efficient training. Figure 1 shows the taxonomy of representative approaches.

3.1 GAN architecture

When GAN was first introduced to the field of machine learning, training GAN was known to be extremely difficult. Dynamics between the generator and the discriminator often undergo a mode-collapse problem for unknown reasons, resulting in meaningless outputs. The first prominent results in GAN development came from adopting a deep convolutional neural network as a backbone architecture (DCGAN) [2] instead of multi-layer perceptron [1]. Although the quality is still far from being realistic, DCGAN is at least capable of producing recognizable images and significantly stabilizes the dynamics between the generator and discriminator. As deep learning and computing acceleration technologies advance, GAN benefits from larger and more complex network architectures [7], [8], [9], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64]. Gulrajani *et al.* [52] develop a ResNet-style generator and discriminator. Zhang *et al.* [49] show that adding self-attention layers into GAN can improve image generation performance (SAGAN). Brock *et al.* [7] scale up the ResNet-style architecture with hierarchical embedding (BigGAN and BigGAN-Deep). Apart from the above stream, Karras *et al.* have proposed a series of GAN architectures starting from Progressive GAN [53], to StyleGAN [54], StyleGAN2 [8], and StyleGAN3 [9]. The StyleGAN family specializes in generating images with a low inter-class variation using more inductive biases in the architecture structure than SAGAN and BigGAN. Beyond the narrow distribution modeling capabilities of StyleGAN-family models, Kang *et al.* [64] and Sauer *et al.* [65] develop GigaGAN and StyleGAN-T, featuring much bigger model capacity, and demonstrate GAN’s potential for a more generic image synthesis task, *i.e.*, text-to-image synthesis.

3.2 Conditional image generation

Conditional image generation is a task to synthesize an intended image provided that conditional information is given. The conditional information could be an image [12], [14], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], a categorical label [10], [27], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], or a text [88], [89], [90], [91], [92], [93]. With GAN’s surging popularity, conditional GAN (cGAN) has become a representative framework for conditional image generation, and the cGANs specialized in each conditional information are being studied as separate research topics.

The majority of conditioning methods for the generator are conducted by conditional batch normalization (cBN) [94], [95] or simple concatenation operation. Unlike the generator, the conditioning techniques for the discriminator are developed based on objective functions accompanied by a slight modification of the architecture. ACGAN [79] adopts an auxiliary classifier on top of the discriminator and tries to minimize the adversarial loss and classification objective together. PD-GAN (Projection Discriminator GAN) [81] combines projected categorical embeddings with discriminator’s embeddings, so adversarial training is performed both on the discriminative process and on the categorical classification. ReACGAN [10] succeeds in the spirit of ACGAN and adopts data-to-data cross-entropy (D2D-CE) loss for class conditioning. The primary motivation for ReACGAN invention is ACGAN’s gradient exploding problem, and the exploding issue can be significantly alleviated by normalizing the discriminator’s features onto a unit hypersphere.

3.3 Adversarial losses and regularizations

The unstable nature of GAN dynamics is a major obstacle in high-quality image generation. To unearth the source of the instability, many researchers strive to provide interpretations of the unstable training issue and propose

prescriptions in the form of adversarial loss and regularization. Arjovsky *et al.* [96], [97] state that this unstable characteristic attributes to the inherent gradient vanishing problem of the vanilla GAN loss. According to their papers, a vanilla GAN, which tries to minimize the Jensen-Shannon divergence between real and fake distributions, cannot provide helpful gradient signals to the discriminator if the distribution of fake images is far away from the real data distribution. To detour the gradient vanishing problem, Arjovsky *et al.* propose WGAN that minimizes 1-Wasserstein distance instead of the Jessen-Shannon criterion. Since training WGAN requires restricting the discriminator to a 1-Lipschitz function, several WGAN variants are proposed with the change of imposing the restriction: WGAN-GP [52], WGAN-DRA [98], and WGAN-LP [99]. On the other side, GANs based on Integral Probability Measure (IPM) [100] are proposed by many researchers. WGAN [96], McGAN [101], MMD GAN [102], Fisher GAN [103], Cramér GAN [104], and SphereGAN [105] are representative IPM-based GANs that require a restricted discriminator architecture. However, those models cannot produce successful results on large-scale image datasets, such as ImageNet [106]. Meanwhile, Miyato *et al.* [107] show successful results in ImageNet generation by devising spectral normalization, which bounds the gradient w.r.t an input image to 1.0. Consequently, modern GANs can be divided into two streams: (1) SNGAN [107], SAGAN [49], BigGAN [7], and ReAC-GAN [10] that require spectral normalization for stable training and (2) StyleGAN2 [8], and StyleGAN3 [9] that use R1 regularization [108] which also shrinks the norm of gradient significantly.

3.4 Techniques for data-efficient Training

Brock *et al.* [7] found that the discriminator of GAN tends to memorize the training dataset and provide an authenticity score for a given image, disregarding the realism of the generated images. This observation implies that if the training dataset is not enough, the discriminator can quickly memorize the training dataset, accelerating the training collapse. To this empirical discovery, data-efficient training has become one of the popular research trends in the GAN community. Researchers apply data augmentations on real and fake images to prevent the discriminator’s overfitting. Differentiable augmentation (DiffAug) [109], adaptive discriminator augmentation (ADA) [110], and adaptive pseudo augmentation (APA) [111] are some notable studies that deal with data augmentations for data-efficient learning. Unlike the augmentation-based approaches, Tseng *et al.* presents LeCam regularization [112], which transforms the WGAN training procedure to minimize LeCam divergence under mild assumptions. Since LeCam divergence is known to be more robust under the limited data situation than other f -divergence variants, GAN-imposed LeCam is expected to be data-efficient.

4 INAPPROPRIATE PRACTICES

GAN approaches have been improved remarkably. However, there are inappropriate practices in data preprocessing and evaluation of GAN that can be boiled down to four points as follows:

- Adopting an improper image resizer and applying an unnecessary quantization process during the preprocessing step. (Sec. 4.1)
- The image resizer applied before the evaluation backbone (Feature Extractor, F) is not the same as the resizer used for training the evaluation backbone. (Sec. 4.2)
- Evaluating GAN only with IS and FID that are based on the pre-trained InceptionV3. (Sec. 4.3 and Sec. 4.4)
- Lacking the description of the type of normalization operation in batch normalization layers, although it can significantly affect the results. (Sec. 4.5)

Following the conclusions from these sections, we suggest protocols for training and evaluating GANs in Sec. 5.

4.1 When processing training images

The improper practices in preprocessing image data can be summarized into three points: (1) inherent aliasing artifacts in the training dataset due to improper resizing functions, (2) sampling error occurred by adopting a poor interpolation filter for both up- and down-sampling, and (3) quantization error that raises when converting a training dataset into other data I/O formats (*i.e.*, HDF5).

We highlight less attention to avoiding aliasing artifacts in training images. As Parmar *et al.* [29] and Karras *et al.* stated [9], processing training dataset using not only anti-aliasing but also a high-quality resizer is necessary for realistic image generation. However, most GAN approaches do not report what kind of resizer is used to preprocess images. By investigating original implementations, we identify that many methods use a **bilinear interpolation** for the image resizer, so resized images are occasionally of poor quality and exhibit aliasing artifacts. Furthermore, some studies utilize Tiny-ImageNet [113] dataset which contains aliasing and noise artifacts (Figure A1).

This work proposes to use either **bicubic** [114] or **Lanczos** [115] **interpolation** for image up- and down-sampling to preprocess images. Interestingly, Parmar *et al.* [29] discovered that resizers implemented in popular TensorFlow [116], PyTorch [117], and OpenCV [118] libraries could cause aliasing artifacts in resulting images. According to this finding, we recommend using python imaging library (PIL) [119], such as PIL.BICUBIC or PIL.LANCZOS resizer.

Lastly, we discover that some popular approaches have an unnecessary or improper quantization process during the data preprocessing step. For example, software platforms built upon PyTorch-BigGAN [120] convert an image from PNG or JPEG to HDF5 format for fast I/O. The software platforms perform unnecessary normalization and wrong quantization operations during the conversion, raising quantization errors. Practitioners should skip the unnecessary normalization process and use a proper quantization operation to avoid the quantization error.

For example, if we use NumPy [121] framework for uint8 quantization, it is necessary to add 0.5 to each pixel since NumPy adopts the round-down operation for uint8 quantization. We express one of the possible solutions for image quantization in the NumPy environment as follows:

$$\mathbf{x}_q = \text{clip}(127.5\mathbf{x}_n + 128, 0, 255), \quad (4)$$

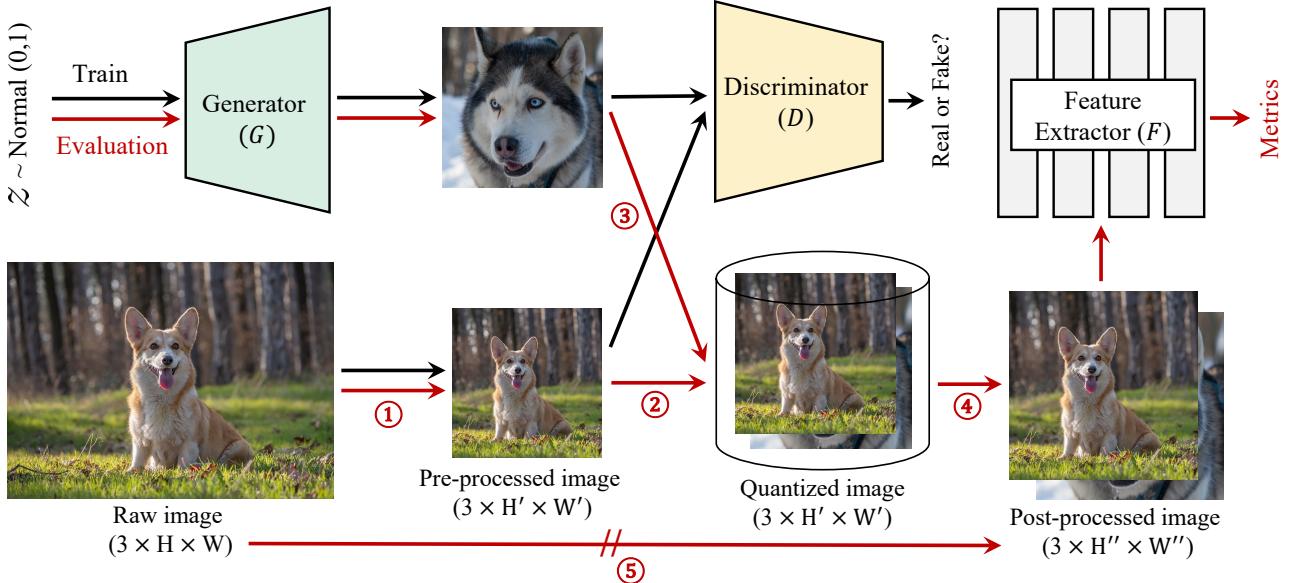


Figure 2. The schematic describes how real and fake images are processed during training and evaluation. Resizing and normalization operations are applied to the raw image in the ① procedure. Then, the pre-processed and generated images are fed into the discriminator to perform adversarial training. When evaluating GAN, it is recommended to quantize both real and generated images to 8-bit representation to save images into a disk (② and ③) as stated in [29]. Finally, the quantized images are resized and normalized before being fed to the evaluation backbone (④). We suggest using an architecture-friendly anti-aliasing resizer instead of directly adopting a high-quality resizing filter. Note that skipping the procedures ① and ② and processing the raw image using ⑤ procedure should not be allowed since it will make a discrepancy between the generator’s ideal distribution and the real data distribution so leading to biased evaluation.

where $\text{clip}(\cdot, \text{lower bound}, \text{upper bound})$ is a clipping function, x_q is an image quantized in unit8 type, and x_n is a normalized image with a pixel range of [-1.0, 1.0].

4.2 When processing generated images

We observe another hazard during the GAN evaluation. It attributes to the choice of image resizer applied to the generated images when feeding them to the evaluation backbone (*i.e.*, TensorFlow-InceptionV3 to acquire FID and IS scores). Regarding this issue, Parmar *et al.* [29] confirmed that the improper resizing operation provokes aliasing artifacts, and they suggest clean-metric (clean-FID) by using the PIL.BICUBIC resizer [119].

In this work, we suggest using **backbone friendly resizer** for the evaluation. Our key finding is that the proper image resizer depends on the backbone network. For instance, feeding bicubic-interpolated images to the InceptionV3 network is improper since the InceptionV3 network was originally trained using a different resizer. Adopting the same resize filter used in training the TensorFlow InceptionV3 network cannot be the solution. This is because the InceptionV3 network was trained using the direct route ⑤ as depicted in Figure 2 while real images for GAN should be resized using the anti-aliasing bicubic/Lanczos resizer through the route ①. This fact makes the entire data processing procedures of a recognition model and GAN inevitably different. Also, using a resizer implemented in TensorFlow for ⑤ method can induce aliasing artifacts as discovered by Parmar *et al.* [29], so both generated and real images may lose details. For that matter, we recommend using a resizer in the PIL library as suggested by Parmar *et al.* [29], but with a different interpolation filter applied.

To identify which resizer is more suitable for the specific backbone, we experiment with the assumption that the re-

sizer is better if it allows better image classification accuracy. If the evaluation backbone shows better recognition ability, the backbone is likely to extract more disentangled features. The assumption correlates with the reason why InceptionV3 network is widely applied as an evaluation backbone network. The authors of IS [27] selected InceptionV3 [30] as a feature extractor since it was one of the cutting-edge models at that time.

In Table 1, we measure Inception Score [27], Top-1, and Top-5 image classification accuracies on ImageNet [106] validation dataset with various image resizers applied for the procedure ④ in Figure 2. Interestingly, the results are sensitive depending on the choice of image resizer, which indicates GAN comparison should be carefully designed. Experimental results also show that PIL.BILINEAR exhibits the coherent Inception Score (IS) and recognition accuracies in most cases for TensorFlow-InceptionV3 [30]. To see the effects of image resizer in modern backbone networks, we perform the same experiment with PyTorch-SwAV [34] and PyTorch-Swin-T [37]. In the case of PyTorch-SwAV, PIL.BILINEAR is considered a reasonable choice. One worth mentioning point is that processed images using PIL.NEAREST resizer gives exceptionally high Swin Transformer Scores (TS). For this reason, we weight Top-1 and Top-5 accuracies to select the proper resizer for Swin-T and decide to use PIL.BICUBIC resizer.

4.3 Backbone networks for GAN evaluation

Conventionally, almost all scores for generative model evaluation are computed using the features extracted from the TensorFlow-InceptionV3 network. Since Fréchet Distance (FD) measured using TensorFlow-InceptionV3 network (FID) is sample-inefficient [35] and is biased towards ImageNet classification task [31], it sometimes does not cap-

TABLE 1. Experimental results to check IS [27] and classification accuracies of an evaluation backbone according to the resizer type. We use ImageNet [106] valid dataset to measure the metrics. We resize a raw image using PIL.LANCZOS resizer (procedure ①) and quantize the image (procedure ②). The numbers shown on the left side of the table (128, 256, and 512) indicate the resolution of the quantized image, and each image is resized once again to adjust the resolution for the evaluation backbone (procedure ④). The final resolution depends on the types of evaluation backbones: 299 for TensorFlow-InceptionV3 [30], 224 for PyTorch-SwAV [34], and 224 for PyTorch-Swin-T [37]. We measure Inception Score (IS), SwAV Score (SS), Swin-Transformer Score (TS), Top-1 accuracy, and Top-5 accuracy. SS and TS have the same conceptual scores as IS, except that the InceptionV3 backbone is replaced with SwAV and Swin-T, respectively. The numbers in bold-faced denote the best performances, and yellow colored blocks indicate architecture-friendly resizers for each evaluation backbone.

| RESIZER ④ | TensorFlow-InceptionV3 [30] | | | PyTorch-SwAV [34] | | | PyTorch-Swin-T [37] | | | |
|-----------|-----------------------------|---------------|--------------|-------------------|---------------|--------------|---------------------|---------------|--------------|--------------|
| | IS ↑ | TOP-1 ↑ | TOP-5 ↑ | SS ↑ | TOP-1 ↑ | TOP-5 ↑ | TS ↑ | TOP-1 ↑ | TOP-5 ↑ | |
| 128 | NEAREST | 167.06 | 67.88 | 88.07 | 153.46 | 59.71 | 82.80 | 106.32 | 75.35 | 92.79 |
| | BILINEAR | 182.57 | 72.03 | 90.09 | 276.67 | 69.78 | 89.46 | 61.88 | 78.21 | 94.81 |
| | BICUBIC [114] | 178.45 | 71.84 | 89.80 | 273.76 | 69.26 | 88.99 | 71.70 | 78.93 | 95.16 |
| | LANCZOS [115] | 173.79 | 71.42 | 89.53 | 254.81 | 68.36 | 88.37 | 74.55 | 79.65 | 95.27 |
| 256 | NEAREST | 260.17 | 77.65 | 93.98 | 299.90 | 72.79 | 91.15 | 213.70 | 84.87 | 97.45 |
| | BILINEAR | 259.55 | 78.11 | 94.08 | 327.90 | 74.39 | 92.06 | 154.52 | 84.14 | 97.13 |
| | BICUBIC [114] | 259.99 | 78.26 | 94.16 | 321.78 | 74.04 | 91.85 | 194.74 | 85.13 | 97.51 |
| | LANCZOS [115] | 259.31 | 78.27 | 94.13 | 316.78 | 73.68 | 91.64 | 202.29 | 85.00 | 97.39 |
| 512 | NEAREST | 262.30 | 77.89 | 94.00 | 302.56 | 72.57 | 91.22 | 212.28 | 84.51 | 97.29 |
| | BILINEAR | 265.22 | 78.48 | 94.30 | 341.60 | 74.66 | 92.25 | 156.38 | 84.28 | 97.18 |
| | BICUBIC [114] | 265.14 | 78.44 | 94.30 | 334.83 | 74.34 | 92.04 | 192.10 | 85.12 | 97.48 |
| | LANCZOS [115] | 264.32 | 78.41 | 94.26 | 326.64 | 73.99 | 91.78 | 201.03 | 85.12 | 97.44 |

ture the discrepancy between distributions well. Therefore, we investigate GAN evaluation using a modern recognition backbone (Swin-T [37]) and self-supervised backbones (SwAV [34] and DINO [36]).

Let \mathcal{X}_S and \mathcal{X}_T be the source and target datasets, and $\mathcal{X}_t \subset \mathcal{X}_T$ be a subset of the target dataset. Then, **relative-FD** can be written as $\frac{\text{FD}(\mathcal{X}_S, \mathcal{X}_t, F)}{\text{FD}(\mathcal{X}_S, \mathcal{X}_T, F)}$ to analyze the sample efficiency of FD metrics across different backbones F . Figure 3 shows relative-FD computed using pre-trained BigGAN models [7] on CIFAR10 [122] and ImageNet [106]. Lower is better because it indicates we can evaluate small number of generated images reasonably. The figure implies that the evaluated FDs using the InceptionV3 network are sample-inefficient compared to the other networks, as stated in [123]. Note that SwAV, DINO, and Swin-T give relatively sample-efficient results on CIFAR10 and ImageNet datasets.

To explore the effectiveness of FD, we visualize trends of **real-to-real-FDs** as the number of samples increases in Figure 4. Using various backbone networks, we measure the feature distance between real images and the fraction of real images. Using the evaluation backbones, we calculate real images' moments (mean and variance). Then, we sample fractions of the real images and calculate FDs. The real-to-real-FD can be written as $\text{FD}(\mathcal{X}_S, \mathcal{X}_s, F)$, where $\mathcal{X}_s \subset \mathcal{X}_S$. Unlike the results in Figure 3, Figure 4 shows that DINO does not measure the real-to-real-FD efficiently. We speculate this phenomenon attributes to the property of DINO capturing the layout of a scene. As a result, we believe that **SwAV**, and **Swin-T** are good backbone candidates for GAN evaluation in addition to the InceptionV3 network.

4.4 Evaluation metrics

Currently, GAN evaluation hugely depends on the value of Fréchet Inception Distance (FID) [28]. While FID has been verified by subsequent papers [7], [8], [9], [107], it has several limitations that FID informs the general ability of a generative model instead of the model's specialties [32], [33], [124]. Also, FID is sometimes not coherent with human

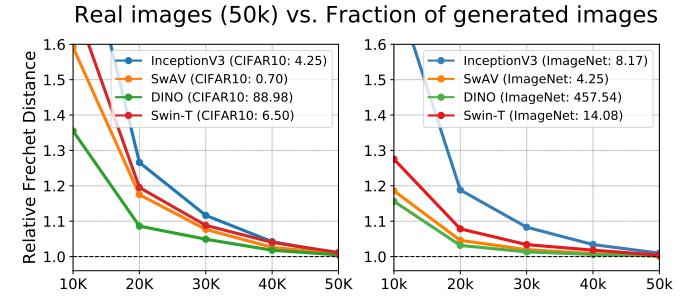


Figure 3. Trends of relative Fréchet Distance (FD) between real images (50k) and a fraction of generated images. To calculate the target FD, we sample 50k generated images using BigGAN trained using StudioGAN, and the computed FD is used as the reference for calculating the relative FD. The reference values are written in the legends. The figures show how efficiently each backbone can measure FD.

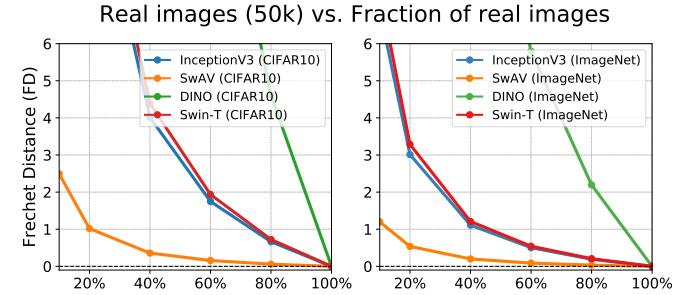


Figure 4. Trends of Fréchet Distance (FD) between real images (50k) and a fraction of the real images. The Figures show how efficiently each backbone can measure real-to-real-FD.

judgment [35], [60], [125]. To complement FID, Precision & Recall metrics [32], [33], [124] are proposed. However, those metrics are not as widely used as FID, and the criterion for computing the metrics has not been established concretely [63]. Therefore, we emphasize that evaluating GAN using various metrics with concrete criteria is necessary to analyze the various aspect of GAN.

TABLE 2. Experiments to check whether the normalization operation of batch normalization layers can affect the evaluation results. We utilize a pre-trained BigGAN [7] and evaluate the model by changing a normalization operation.

| | Normalization | IS ↑ | FID ↓ | Precision ↑ | Recall ↑ |
|---------|-------------------------|--------|-------|-------------|----------|
| CIFAR10 | Moving average [126] | 9.92 | 4.65 | 0.74 | 0.64 |
| | Batch statistics [126] | 9.93 | 4.12 | 0.74 | 0.65 |
| | Standing statistics [7] | 9.96 | 4.16 | 0.74 | 0.65 |
| ImgNet | Moving average [126] | 100.69 | 10.23 | 0.74 | 0.58 |
| | Batch statistics [126] | 87.62 | 10.86 | 0.73 | 0.60 |
| | Standing statistics [7] | 98.51 | 8.54 | 0.75 | 0.60 |

4.5 Normalization operations

We experimentally discover that the normalization operation of batch normalization layers can significantly affect the generation quality in testing time if the generator contains a batch normalization layer. Specifically, the generation results vary depending on the way of calculating statistics (mean and variance of features) for the normalization operation: (1) using updated moving average statistics [126], (2) adopting batch statistics [126], and (3) standing statistics [7]. The moving average statistics are computed as the moving average of batch statistics during training time, exactly how the batch normalization layer works in testing time. On the other hand, batch statistics work in the same fashion as batch normalization in training time. Unlike the previous two approaches, standing statistics proposed by Brock *et al.* [7] compute the statistics by accumulating statistics of multiple batches (*number of repetition*) whose batch size is randomly determined from 1 to *maximum batch size*. As shown in Table 2, (1) and (2) give poor synthesis outputs compared to (3) based on various metrics, including IS, FID, and Precision & Recall. In addition, when we evaluate GAN using (2), the generation performance differs according to the batch size, resulting in inconsistent evaluation. We suggest using the **standing statistics** for GAN evaluation.

4.6 Reproducibility

In addition to the previous issues, we observe that open-sourced GANs have some mistakes and missing statements. For instance, when evaluating a generative model during training for monitoring purposes, it is necessary to switch the generative model to the evaluation mode that freezes the moving average moments in the batch normalization layers. This prevents the model from cheating information in the validation dataset. In addition, many GAN approaches do not mention which dataset split is used for the evaluation. The lack of description induces misleading comparison since the metrics are sensitive to the reference distribution. Therefore, GAN approaches should clarify the dataset type (train, validation, and test) and the number of reference data for a fair evaluation. If a subset of a dataset is used in the assessment, providing the subset is required for reproducibility.

Moreover, training results dramatically change depending on the choice of hyperparameters, hidden details, and even the specific release version of machine learning framework (Caffe [127], TensorFlow [116], PyTorch [117], *etc.*). As a result, it is hard to reproduce the reported performance of GAN even with the authors' official implementation.

For example, the authors' implementation of BigGAN [120] is not guaranteed to reproduce the experiment on ImageNet dataset [128], [129]. Moreover, image generation benchmarks on CIFAR10 [122] and ImageNet [106] are not compared carefully, so the reported performances in BigGAN [7], TACGAN [83], OmniGAN [86], ReACGAN [10], and ADCGAN [87] are not consistent, resulting in an unfair and improper comparison between models.

5 PROTOCOL SUGGESTION

With the discussions and observations in Sec. 4, it is obvious to build a standard protocol before making a fair comparison. Therefore, we propose the training and evaluation protocols to train and evaluate a variety of GAN models.

Training protocol. When training a GAN model for high-quality image generation, it is important to keep the quality of training images as good as possible:

- 1) If an image resizer is required in training, use a high-quality resizer (*e.g.*, PIL.BICUBIC and PIL.LANCZOS) that minimizes aliasing artifacts.
- 2) If there is a need to save processed images into a disk, keep the images with lossless compression (*e.g.* png) or with little lossy compression (*e.g.* high-quality jpeg).
- 3) Avoid applying any unnecessary quantization operation to the training dataset.
- 4) When evaluating a generative model during training, switch the generative model to freeze the batch normalization layers in the generative model.

Evaluation protocol. When evaluating a GAN model, reproducibility and fairness are the crucial factors. Therefore, we present an evaluation protocol as follows:

- 1) Save generated images in the lossless png format to a disk using a proper image quantization operation.
- 2) Resize real and fake images using the backbone-friendly resizer described in section 4.2.
- 3) Evaluation of a generative model should include FD (*e.g.* FID) and metrics that can quantify fidelity (*e.g.* Precision) and diversity (*e.g.* Recall) of generated samples [32], [33].
- 4) If the generator contains batch normalization layers, apply the standing statistics technique proposed by [7].
- 5) Clarify the dataset name and split (training, validation, and test) and the number of reference data for a fair evaluation.
- 6) Specify the version of the deep learning framework and the included library to reproduce the results.

6 STUDIOGAN

This section introduces StudioGAN, a software library that provides reproducible implementations of representative GANs. StudioGAN is motivated by practical needs, where reproducibility has been a ball and chain around the ankle of GAN development. StudioGAN supports over **30 popular GANs** and **GAN-related modules** based on the PyTorch framework (refer to the appendix for the full list).

In short, StudioGAN has the following key features:

TABLE 3. We check the reproducibility of GANs implemented in StudioGAN by comparing IS [27] and FID [28] with the original papers. We identify our platform successfully reproduces most of representative GANs except for PD-GAN [81], ACGAN [79], LOGAN [133], SAGAN [49], and BigGAN-Deep [7]. FQ means Flickr-Faces-HQ Dataset (FFHQ) [54]. The resolutions of ImageNet [106], AFHQv2 [9], [75], and FQ datasets are 128, 512, and 1024, respectively.

| Method | Paper | | Our StudioGAN Implementation | |
|---------------|-----------------------------|------------------|------------------------------|-----------------------------|
| | IS \uparrow | FID \downarrow | IS \uparrow | FID \downarrow |
| CIFAR10 | WGAN-GP [52], [107] | 6.68 | 40.1 | 6.71 (\uparrow 0.03) |
| | PD-GAN [81] | 8.62 | 17.5 | 7.26 (\downarrow 1.36) |
| | SNDGAN [107] | 7.42 | 29.3 | 8.20 (\uparrow 0.78) |
| | SNResGAN [107] | 8.22 | 21.7 | 8.98 (\uparrow 0.76) |
| | BigGAN [7] | 9.22 | 14.73 | 9.97 (\uparrow 0.75) |
| | StyleGAN2 [8] | 9.53 | 6.96 | 10.28 (\uparrow 0.75) |
| | MHGAN [84], [110] | 9.58 | 6.40 | 10.13 (\uparrow 0.55) |
| | StyleGAN2 + ADA [110] | 10.14 | 2.42 | 10.46 (\uparrow 0.32) |
| | CRGAN [38] | - | 11.48 | 10.38 |
| | ICRGAN [134] | - | 9.21 | 10.15 |
| ImageNet | BigGAN-DiffAug [109] | 9.25 | 8.59 | 9.78 (\uparrow 0.53) |
| | BigGAN-LeCam [112] | 9.31 | 8.31 | 10.13 (\uparrow 0.82) |
| | ACGAN [79], [80] | 8.25 | - | 7.33 (\downarrow 0.92) |
| | LOGAN [133] | 8.67 | 17.7 | 7.66 (\downarrow 1.01) |
| | SNGAN [107], [135] | 36.80 | 27.62 | 32.40 (\downarrow 4.4) |
| AFHQv2 | SAGAN [49] | 51.43 | 18.28 | 19.16 (\downarrow 32.27) |
| | BigGAN (B.S.=2048) [7] | 98.8 | 8.7 | 97.76 (\downarrow 1.04) |
| | StyleGAN3-t [63] | 15.30 | 53.57 | 20.99 (\uparrow 5.69) |
| | BigGAN (B.S.=256) [7], [83] | 38.05 | 22.77 | 43.97 (\uparrow 5.92) |
| | StyleGAN2 + ADA [110] | - | 4.62 | 12.13 |
| FQ | StyleGAN3-t + ADA [9] | - | 4.04 | 11.84 |
| | StyleGAN3-r + ADA [9] | - | 4.40 | 11.97 |
| StyleGAN2 [8] | | - | 2.70 | 5.17 |
| | | - | - | 2.76 (\uparrow 0.06) |

- **Coverage:** StudioGAN is a self-contained library that provides 7 GAN architectures, 9 conditioning methods, 4 adversarial losses, 13 regularization modules, 3 augmentation modules, 8 evaluation metrics, and 5 evaluation backbones. Among these configurations, we formulate 30 GANs as representatives.
- **Flexibility:** Each modularized option is managed through a configuration system that works through a YAML file, so users can train a large combination of GANs by mix-matching distinct options.
- **Reproducibility:** With StudioGAN, users can compare and debug various GANs with the unified computing environment without concerning about hidden details and tricks. Table 3 shows that StudioGAN successfully reproduces the most representative GANs on four benchmark datasets.
- **Versatility:** StudioGAN supports 5 types of acceleration methods with synchronized batch normalization for training: a single GPU training, data-parallel training (DP), distributed data-parallel training (DDP), multi-node distributed data-parallel training (MDDP), and mixed-precision training [130].

The implemented methods in StudioGAN are marked with colored boxes in Figure 1. We gently note that StudioGAN is not the only attempt to provide various implementations. Repositories such as MMGeneration [131] and Mimicry [132] provide 16 and 6 GAN implementations, respectively, with IS and FID values. While these platforms are beneficial, they evaluate GAN with limited metrics and have little flexibility to mix-match diverse backbones and recent training techniques.

7 BENCHMARK

7.1 Reproducibility of StudioGAN

Before presenting a benchmark, we examine whether our software library can reproduce evaluation results of well-known GAN papers. We utilize CIFAR10, ImageNet, AFHQv2, and FFHQ (FQ) datasets, along with Inception Score (IS) and Fréchet Inception Distance (FID), for the reproducibility check. StudioGAN generally presents superior image synthesis results in most cases, with the exceptions of PD-GAN [81], ACGAN [79], LOGAN [133], and SAGAN [49]. Investigating the performance discrepancies between original implementations and StudioGAN would give researchers insights into successful GAN training; however, pinpointing exact causes is extremely demanding, as implemented libraries, used datasets, and minor details vary from paper to paper. For example, StudioGAN seems to successfully replicate the results of SNGAN [107] on ImageNet but struggles to reproduce the results of SAGAN [49]. The only differences between SNGAN and SAGAN are the self-attention layers [136] and changed learning rate as suggested by the SAGAN paper [49]. We notice such small difference results in the reproducibility issue.

BigGAN [7] and StyleGAN-family models [8], [9], [54], [63], [110] have become the standard for GAN developments. Since BigGAN and StyleGAN-family models are open sourced by the authors, IS [27] and FID [28] of these models have shown continuous improvement. Regarding StyleGAN2 on CIFAR10 and ImageNet, we discover that the performance discrepancies between StudioGAN and the original implementation are due to the difference in data pre-processing methods. Despite our efforts to match model parameters, hyperparameter settings, and data processing setup, some results still differ from original implementations. We attribute this to slight differences in low-level implementation details. From our experience, even minor changes, such as loading sequence of tensors to CUDA, can lead to considerable variations in training results. For BigGAN, FID scores on CIFAR10 are different across papers (14.73 in [7], 7.27 in [86], and 4.16 in our paper). We speculate that the superior performance of BigGAN from StudioGAN can be attributed to the different hyperparameter settings and the fine-details mentioned above as well as the employment of standing statistics [7], and synchronized batch normalization [137].

7.2 Baselines

We provide a comprehensive GAN benchmark using the StudioGAN library. We train and evaluate GANs spanning primitive DCGAN [2] to recent StyleGAN3 [9], built upon four different types of backbone for the generator and the discriminator (DCGAN, ResNetGAN, BigGAN, and StyleGAN). Some GANs (such as CRGAN [38] and DiffAug [109]) can be applied to multiple GAN backbones, but we use the same generator and discriminator architectures described in the original paper to provide evaluation results close to the original study.

BigGAN and StyleGANs are representative GANs for natural image synthesis. While those models are being actively studied, BigGAN and StyleGANs tend to be studied separately in the literature because StyleGANs are designed

for unconditional image generation, whereas BigGAN is designed for conditional image generation. To compare those models in a fair way, we adopt the class conditioning version of StyleGAN2 [110], StyleGAN3-t [9], and StyleGAN3-r [9] for all experiments except for the cases on AFHQv2 and FFHQ datasets in Tables 5, A3, and A4. Because of the poor convergence issue, StyleGAN3-r results on ImageNet are omitted from Tables 5, A3, and A4. Here, we utilize InceptionV3, SwAV, and Swin-T for the evaluation backbone to avoid biased conclusions.

7.3 Datasets

We use four standard datasets for the conditional image generation: **CIFAR10** [122], **ImageNet** [106], **AFHQv2** [9], [75], and **FFHQ** [54] and three of our proposed ImageNet subsets: **Baby/Papa/Grandpa-ImageNet**. We create the subsets of ImageNet for a small-scale ImageNet experiment because training GAN on the full ImageNet requires extensive computational resources. The proposed subsets of the ImageNet are made according to the classification difficulty, enabling researchers to analyze behaviors of conditional GANs based on the level of classification difficulty. Baby-ImageNet contains the easiest images to classify, whereas the Grandpa-ImageNet contains the hardest ones. We attach a detailed explanation of datasets in the appendix. Tiny-ImageNet [113] is also intended for the small-scale ImageNet experiment, but we decided not to use it for the benchmark because Tiny-ImageNet is known to have severe aliasing artifacts and degradation.

7.4 Evaluation metrics

We utilize 7 evaluation metrics: Inception Score (IS) [27] (Eq. 2), Fréchet Inception Distance (FID) [28] (Eq. 3), Precision & Recall [32], Density & Coverage [33], and Intra-class FID (IFID) [49] that is the average of class-wise FID. Each metric is computed using TensorFlow-InceptionV3 network in which weights are borrowed from Improved-GAN GitHub [138]. We use PyTorch implementations of IS (we implemented), FID [139], Precision & Recall [140], and Density & Coverage [140] to construct seamless training and evaluation procedures.

Precision & Recall are used to quantify the fidelity and diversity of generated images based on the estimated support of real and fake distributions. Precision is defined as the portion of randomly generated images that falls within the support of real data distribution. On the other hand, Recall is the portion of real images falling within the support of fake data distribution. High Precision and Recall mean that the generated images are high-quality and diverse. Mathematically, Precision & Recall can be written as follows:

$$\text{Precision}(\mathbf{X}_s, \mathbf{X}_t, F, k) := \frac{1}{M} \sum_{i=1}^M \mathbb{I}\left(\mathbf{f}_{t,i} \in \mathcal{M}(F(\mathbf{X}_s), k)\right), \quad (5)$$

$$\text{Recall}(\mathbf{X}_s, \mathbf{X}_t, F, k) := \frac{1}{N} \sum_{i=1}^N \mathbb{I}\left(\mathbf{f}_{s,i} \in \mathcal{M}(F(\mathbf{X}_t), k)\right), \quad (6)$$

where $\mathcal{M}(F, k) := \bigcup_{\mathbf{f} \in F} B(\mathbf{f}, \text{NN}_k(F, \mathbf{f}, k))$, F is a collection of features from F , $\mathbf{X}_s = \{\mathbf{x}_{s,1}, \dots, \mathbf{x}_{s,N}\}$ is a source

TABLE 4. Metric names w.r.t. various backbone networks.

| Backbone | Score | Fréchet Distance | Precision | Recall | Density | Coverage |
|------------------|-------|------------------|-------------|----------|-----------|------------|
| InceptionV3 [30] | IS | FID | Precision | Recall | Density | Coverage |
| SwAV [34] | SS | FSD | S-Precision | S-Recall | S-Density | S-Coverage |
| Swin-T [36] | TS | FTD | T-Precision | T-Recall | T-Density | T-Coverage |

dataset (real images), $\mathbf{X}_t = \{\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,M}\}$ is a target dataset (generated images), $B(\mathbf{f}, r)$ is the n-dimensional sphere in which $\mathbf{f} = F(\mathbf{x}) \in \mathbb{R}^n$ is the center and r is the radius, $\text{NN}_k(F, \mathbf{f}, k)$ is the distance from \mathbf{f} to the k -th nearest embedding in F , and $\mathbb{I}(\cdot)$ is a indicator function.

Recently, Naeem *et al.* [33] show that the manifold construction process using the nearest neighbor function is vulnerable to outlier samples, so it often results in an overestimated manifold of the distribution. With the overestimation problem fixed by sample counting, Naeem *et al.* propose **Density** & **Coverage** metrics. Mathematically, these metrics can be expressed as follows:

$$\text{Density}(\mathbf{X}_s, \mathbf{X}_t, F, k) :=$$

$$\frac{1}{kM} \sum_{j=1}^M \sum_{i=1}^N \mathbb{I}\left(\mathbf{f}_{t,j} \in B\left(\mathbf{f}_{s,i}, \text{NN}_k(F(\mathbf{X}_s), \mathbf{f}_{s,i}, k)\right)\right), \quad (8)$$

$$\text{Coverage}(\mathbf{X}_s, \mathbf{X}_t, F, k) :=$$

$$\frac{1}{N} \sum_{i=1}^N \mathbb{I}\left(\exists j \text{ s.t. } \mathbf{f}_{t,j} \in B\left(\mathbf{f}_{s,i}, \text{NN}_k(F(\mathbf{X}_s), \mathbf{f}_{s,i}, k)\right)\right). \quad (9)$$

We perform additional evaluations by replacing the InceptionV3 with a pre-trained SwAV [141] and Swin-T [142] as an alternative evaluation backbone. Table 4 summarizes the names of all metrics w.r.t various backbone networks. Note that such a comprehensive re-evaluation of representative GANs with these metrics is not presented so far.

In addition to the 7 metrics listed above, various other metrics are also considered to evaluate the performance of image synthesis models. Notable metrics include MSE, PSNR, SSIM [143], LPIPS [144], Realism score [32], and GIQA [145]. MSE, PSNR, SSIM, and LPIPS measure the reconstruction difference between ground truth and targeted images. Realism score and GIQA measure how close a given image is located from the true data distribution (i.e., how realistically it has been generated). Consequently, these metrics are less suitable for establishing a benchmark for class-conditional image synthesis, which is the primary objective of this paper. Although CAS [146] can measure class-conditional precision and recall, it requires time-consuming ResNet training. Instead, we evaluate models using improved Precision & Recall and Density & Coverage.

7.5 Findings from the evaluation results

Tables 5, A3, and A4 show quantitative results using InceptionV3, SwAV, and Swin-T, respectively. Thin solid lines are used to group GANs with the similar training backbone (DCGAN, ResNet, BigGAN, and StyleGAN). We also visualize the quantitative results in Figures A3, A4, A5, A6, A7, A8, A9, A10, and A11 to analyze results better. For the details about the experiment setup, please refer to the appendix. We provide our key findings from the experiments below.

TABLE 5. Benchmark table evaluated using **TensorFlow-InceptionV3** [30]. The resolutions of CIFAR10 [122], Baby/Papa/Grandpa ImageNet, ImageNet [106], AFHQv2 [9], [75], and FQ [54] datasets are 32, 64, 128, 512, and 1024, respectively. Top-1 and Top-2 performances are indicated in red and blue, respectively.

| TensorFlow-InceptionV3 [30] | | IS \uparrow | FID \downarrow | Precision \uparrow | Recall \uparrow | Density \uparrow | Coverage \uparrow | IFID \downarrow |
|------------------------------------|--------------------------------|---------------|------------------|----------------------|-------------------|--------------------|---------------------|-------------------|
| CIFAR10 | DCGAN [1] | 6.11 | 48.29 | 0.61 | 0.21 | 0.52 | 0.28 | 117.00 |
| | LSGAN [147] | 6.55 | 40.22 | 0.60 | 0.34 | 0.49 | 0.35 | 111.51 |
| | Geometric GAN [148] | 6.61 | 43.16 | 0.58 | 0.25 | 0.48 | 0.31 | 114.59 |
| | ACGAN-Mod [79] | 7.19 | 33.39 | 0.61 | 0.21 | 0.52 | 0.36 | 72.75 |
| | WGAN [97] | 3.59 | 107.68 | 0.45 | 0.02 | 0.29 | 0.08 | 158.11 |
| | DRAGAN [98] | 6.59 | 34.69 | 0.63 | 0.47 | 0.56 | 0.36 | 106.46 |
| | WGAN-GP [52] | 5.22 | 53.98 | 0.68 | 0.26 | 0.67 | 0.27 | 119.22 |
| | PD-GAN [81] | 7.25 | 31.54 | 0.61 | 0.28 | 0.53 | 0.39 | 65.04 |
| | SNGAN [107] | 8.83 | 9.01 | 0.70 | 0.62 | 0.79 | 0.75 | 22.85 |
| | SAGAN [49] | 8.64 | 10.35 | 0.69 | 0.63 | 0.73 | 0.71 | 25.00 |
| ImageNet | TACGAN [83] | 7.68 | 29.51 | 0.62 | 0.28 | 0.57 | 0.41 | 64.52 |
| | LOGAN [133] | 7.89 | 20.45 | 0.65 | 0.63 | 0.61 | 0.54 | 95.82 |
| | BigGAN [7] | 9.96 | 4.16 | 0.74 | 0.65 | 0.98 | 0.88 | 14.29 |
| | CRGAN [38] | 10.09 | 3.16 | 0.74 | 0.66 | 0.97 | 0.90 | 12.90 |
| | MHGAN [84] | 10.07 | 3.95 | 0.73 | 0.65 | 0.92 | 0.88 | 14.42 |
| | ICRGAN [134] | 10.14 | 3.54 | 0.75 | 0.64 | 0.98 | 0.90 | 13.37 |
| | ContraGAN [85] | 9.78 | 6.01 | 0.74 | 0.63 | 0.95 | 0.84 | 119.63 |
| | BigGAN + DiffAug [109] | 10.03 | 3.35 | 0.74 | 0.67 | 0.97 | 0.90 | 12.63 |
| | BigGAN + LeCam [112] | 10.17 | 3.23 | 0.73 | 0.67 | 0.95 | 0.89 | 13.11 |
| | ADCGAN [87] | 9.91 | 3.93 | 0.72 | 0.67 | 0.92 | 0.88 | 14.27 |
| Baby-ImageNet | ReACGAN [10] | 9.85 | 3.87 | 0.75 | 0.62 | 1.01 | 0.88 | 15.17 |
| | StyleGAN2 [8] | 10.17 | 3.78 | 0.72 | 0.67 | 0.97 | 0.89 | 14.04 |
| | StyleGAN2 + DiffAug [8], [109] | 10.55 | 2.39 | 0.74 | 0.68 | 1.02 | 0.92 | 11.56 |
| | StyleGAN2 + ADA [110] | 10.53 | 2.31 | 0.75 | 0.69 | 1.04 | 0.93 | 11.82 |
| | StyleGAN2 + LeCam [8], [112] | 10.04 | 4.65 | 0.71 | 0.66 | 0.90 | 0.86 | 15.66 |
| | StyleGAN2 + APA [111] | 10.05 | 4.04 | 0.70 | 0.68 | 0.89 | 0.87 | 54.40 |
| | StyleGAN2 + D2D-CE [10] | 10.43 | 3.44 | 0.74 | 0.64 | 1.02 | 0.90 | 14.14 |
| | StyleGAN3-r + ADA [9] | 9.72 | 10.83 | 0.68 | 0.53 | 0.84 | 0.75 | 27.32 |
| | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| Papa-ImageNet | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| | SNGAN [107] | 15.04 | 41.06 | 0.52 | 0.55 | 0.38 | 0.37 | 107.13 |
| | SAGAN [49] | 12.86 | 51.55 | 0.48 | 0.53 | 0.31 | 0.27 | 130.10 |
| | BigGAN [7] | 21.62 | 24.99 | 0.60 | 0.47 | 0.54 | 0.53 | 79.35 |
| | ContraGAN [85] | 19.22 | 25.33 | 0.62 | 0.48 | 0.54 | 0.46 | 203.33 |
| | ReACGAN [10] | 21.80 | 21.74 | 0.62 | 0.47 | 0.55 | 0.51 | 91.76 |
| Grandpa-ImageNet | StyleGAN2 [8] | 19.28 | 23.28 | 0.58 | 0.61 | 0.52 | 0.58 | 76.06 |
| | StyleGAN3-t [9] | 9.45 | 88.74 | 0.47 | 0.00 | 0.34 | 0.11 | 234.71 |
| | SNGAN [107] | 12.70 | 41.74 | 0.49 | 0.52 | 0.35 | 0.37 | 110.47 |
| | SAGAN [49] | 11.96 | 50.02 | 0.44 | 0.47 | 0.29 | 0.29 | 131.93 |
| | BigGAN [7] | 17.64 | 24.40 | 0.58 | 0.48 | 0.55 | 0.57 | 78.11 |
| | ContraGAN [85] | 17.86 | 21.14 | 0.64 | 0.43 | 0.64 | 0.54 | 193.93 |
| | ReACGAN [10] | 19.53 | 18.65 | 0.64 | 0.41 | 0.64 | 0.58 | 92.39 |
| | StyleGAN2 [8] | 15.71 | 21.88 | 0.56 | 0.58 | 0.51 | 0.59 | 77.66 |
| | StyleGAN3-t [9] | 7.79 | 93.60 | 0.49 | 0.00 | 0.36 | 0.10 | 243.25 |
| | SNGAN-256 [107] | 31.42 | 31.49 | 0.56 | 0.67 | 0.43 | 0.39 | 112.68 |
| ImageNet | BigGAN-256 [7] | 40.78 | 20.57 | 0.66 | 0.65 | 0.63 | 0.52 | 121.32 |
| | ContraGAN-256 [85] | 25.23 | 29.59 | 0.69 | 0.52 | 0.62 | 0.31 | 162.80 |
| | ReACGAN-256 [10] | 66.67 | 15.65 | 0.77 | 0.38 | 0.88 | 0.53 | 127.31 |
| | StyleGAN2 [8] | 22.54 | 33.40 | 0.54 | 0.63 | 0.40 | 0.34 | 135.19 |
| | StyleGAN3-t [9] | 21.06 | 36.51 | 0.53 | 0.60 | 0.38 | 0.30 | 142.49 |
| | StyleGAN2 [8] | 11.94 | 6.08 | 0.85 | 0.49 | 1.32 | 0.81 | 6.02 |
| | StyleGAN2 + ADA [110] | 11.70 | 4.75 | 0.83 | 0.58 | 1.20 | 0.83 | 4.73 |
| | StyleGAN3-t + ADA [9] | 11.63 | 4.53 | 0.81 | 0.68 | 1.05 | 0.84 | 4.59 |
| | StyleGAN3-r + ADA [9] | 11.74 | 4.62 | 0.81 | 0.66 | 1.04 | 0.84 | 4.55 |
| | StyleGAN2 [8] | 5.27 | 3.16 | 0.77 | 0.60 | 1.09 | 0.90 | - |
| AFHQv2 | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| FQ | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| ImageNet | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| Baby-ImageNet | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| Papa-ImageNet | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| Grandpa-ImageNet | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| ImageNet | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| Baby-ImageNet | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| Papa-ImageNet | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| Grandpa-ImageNet | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| ImageNet | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| Baby-ImageNet | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| Papa-ImageNet | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| Grandpa-ImageNet | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| ImageNet | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| Baby-ImageNet | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| Papa-ImageNet | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| Grandpa-ImageNet | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| ImageNet | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| Baby-ImageNet | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| Papa-ImageNet | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| Grandpa-ImageNet | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| ImageNet | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| Baby-ImageNet | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| Papa-ImageNet | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| | SNGAN [107] | 18.21 | 36.31 | 0.54 | 0.58 | 0.38 | 0.39 | 93.10 |
| Grandpa-ImageNet | SAGAN [49] | 16.47 | 42.10 | 0.50 | 0.53 | 0.33 | 0.32 | 111.11 |
| | BigGAN [7] | 27.47 | 18.64 | 0.63 | 0.47 | 0.59 | 0.57 | 62.22 |
| ImageNet | ContraGAN [85] | 23.14 | 24.67 | 0.62 | 0.47 | 0.53 | 0.45 | 220.72 |
| | ReACGAN [10] | 26.18 | 19.75 | 0.63 | 0.43 | 0.54 | 0.50 | 77.95 |
| Baby-ImageNet | StyleGAN2 [8] | 23.40 | 22.21 | 0.62 | 0.59 | 0.58 | 0.62 | 70.83 |
| | StyleGAN3-t [9] | 9.99 | 102.24 | 0.37 | 0.01 | 0.17 | 0.06 | 235.20 |
| Papa-ImageNet | SNGAN [107] | 18. | | | | | | |

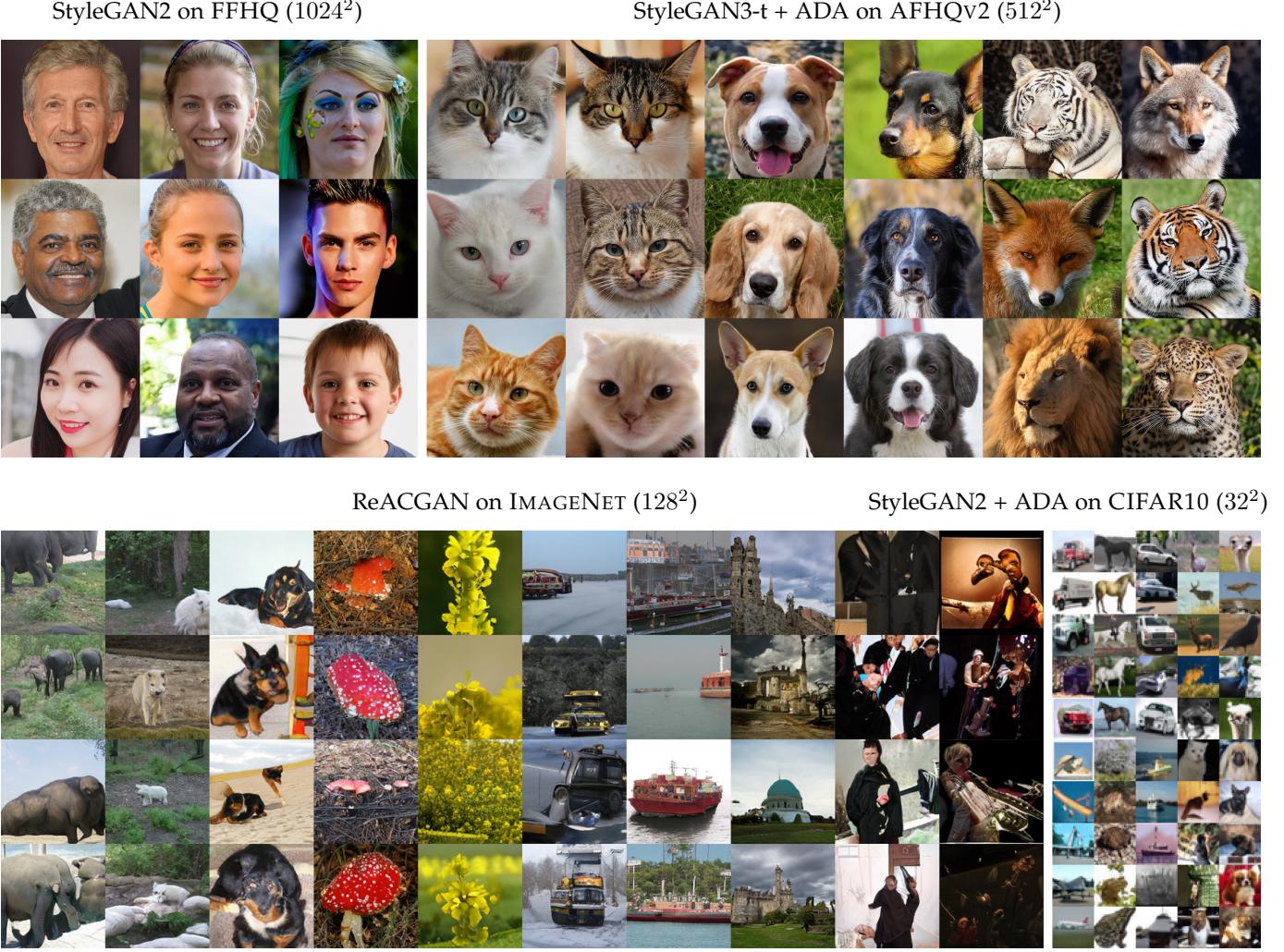


Figure 5. Generated images from GANs implemented in StudioGAN library. We select the best performing GAN on each dataset based on the average rank and generate fake images using those GANs. StyleGAN2 [8] is used for FFHQ [54], StyleGAN3-t + ADA [9] is used for AFHQv2 [9], [75], ReACGAN [10] is used for ImageNet, and StyleGAN2 + ADA [110] is used for CIFAR10 [122].

Rooms for improvement in CIFAR10. CIFAR10 experiment is a starting point for generative model development and has been highly explored by numerous GANs. Interestingly, we find that the generation performances of representative GANs can still be improved with our StudioGAN implementations. Although FID has several drawbacks mentioned in Sec. 4.4, it has long been conventionally adopted as a primary metric for evaluating generative models. Thus we decide to compare FID in this section, as a lot of existing papers only present FID scores. As can be seen in Tables 3 and 5, FID values of SNGAN, BigGAN, StyleGAN2, MHGAN, CRGAN, ICRGAN, and BigGAN-DiffAug are significantly lower than the originally reported numbers. We believe StudioGAN achieves better training dynamics with numerically stable spectral normalization implemented in PyTorch and dedicated hyperparameter selection.

StyleGAN2 can generate more diverse images compared to BigGAN. Experiments using Baby/Papa/Grandpa-ImageNet shown in Table 5 and A3 present that StyleGAN2 models are apt to exhibit higher recall and coverage values than BigGAN counterparts. Figures A9 and A10 in Appendix verify the statement again by showing the orange

markers (StyleGAN2) are located above (high recall and coverage) the red markers (BigGAN). However, based on a few examples, we notice that the recall and coverage values of StyleGAN2 are sometimes lower than those of BigGAN. For example, StyleGAN2 trained using ImageNet shows lower density (BigGAN: 0.65 → StyleGAN2: 0.63) and coverage (BigGAN: 0.52 → StyleGAN2: 0.34) than BigGAN as can be seen in Table 5. We speculate that this is because StyleGAN’s generator design is too restrictive to model complex data distributions like ImageNet.

StyleGAN family struggles with ImageNet. As shown in Tables 5 and A3, StyleGAN2 and StyleGAN3-t present FID values of 33.4 and 36.51 and FSD values of 5.61 and 6.07 on ImageNet generation experiments, respectively, which are worse than those of BigGAN, ContraGAN, and ReACGAN. In contrast, StyleGAN2 has a better synthesis ability than BigGAN (refer to Table A1) with AFHQv2 dataset. Such results imply that StyleGAN2 and StyleGAN3-t are tailored to center-aligned and high-resolution images, and the StyleGAN-family models are not good at generating images having high inter-class variations. This observation becomes more evident when comparing the

FID values of cGAN models trained on subsets of ImageNet (Baby/Papa/Grandpa-ImageNet) and the original ImageNet. StyleGAN2 models trained on Baby-, Papa-, and Grandpa-ImageNet achieve FID scores of 22.21, 23.28, and 21.88, respectively, which are comparable to or even better than those achieved by BigGAN. However, when StyleGAN2 is trained on the full ImageNet, it exhibits a significantly higher FID score (33.40) compared to other cGAN models (BigGAN: 20.57, ContraGAN: 29.59, and ReACGAN: 15.65). This indicates that training conditional StyleGAN2 becomes increasingly challenging as the number of classes in the training data increases.

Fréchet SwAV Distance favors StyleGAN2 over BigGAN. As can be seen in Tables 5 and A3, we discover that FID and FSD show different preferences across BigGAN, ContraGAN, ReACGAN, and StyleGAN2. Specifically, SwAV tends to take the side of StyleGAN2 while InceptionV3 is apt to take the side of BigGAN and ReACGAN. We believe that StyleGAN variants can generate more realistic images because Morozov *et al.* [35] demonstrates that FSD and Precision & Recall measured through SwAV provide more consistent evaluation results with human evaluation. However, given that FID [28] is still broadly adopted, we think that further analysis of FID, FSD, Precision & Recall, and S-Precision & S-Recall is necessary.

Image generation quality vs. image classification accuracy. As shown in Table 5, evaluation results using the InceptionV3 network on Baby/Papa/Grandpa-ImageNet imply no strong correlation between classification difficulty and generation quality. However, Table A3 shows interesting trends that the FSD value of the generated images gets lower if training images are easily classifiable. For instance, FSD values of BigGAN, ContraGAN, ReACGAN, and StyleGAN on Baby-ImageNet are 4.49, 4.47, 4.18, and 3.43, while FSD values of those models on Papa-ImageNet are 5.3, 4.62, 4.3, and 3.7. This relationship holds for Grandpa-ImageNet too. Choosing the most suitable evaluation backbone is an open question, so we do not judge that well-classifiable images are well-generated images. However, we value the correlation between FSD and classification difficulty and believe it is worth exploring in the future.

7.6 Potential hazards

So far, we have described an extensive benchmark based on our newly suggested evaluation protocol. However, as we look over the results and inspect the details of each metric, we found some unclear points and the potential danger of misleading metrics. In this section, we elaborate few potential hazards for GAN community to be aware of in future research.

Evaluation results are highly affected by evaluation backbone. As shown in Tables 5, A3, and A4, the rank of each metric varies as InceptionV3 [30], SwAV [34], or Swin-T [37] is used as an evaluation backbone. InceptionV3 network was adopted as an evaluation backbone because there was a belief that a cutting-edge recognition model is useful for extracting good image features. Still, there is no guarantee that InceptionV3 is the best choice for generative model evaluation. Therefore, we encounter a potential hazard that

TABLE 6. ImageNet classification accuracies on generated images from GANs. We use TensorFlow-InceptionV3 [30] to compute Top-1 accuracy, Top-5 accuracy, and FID.

| | Model | Top-1 acc. \uparrow | Top-5 acc. \uparrow | FID \downarrow |
|----------|----------------|-----------------------|-----------------------|------------------|
| Baby | BigGAN [7] | 53.65 | 72.09 | 18.64 |
| | ContraGAN [85] | 4.44 | 6.52 | 24.67 |
| | ReACGAN [10] | 50.71 | 70.37 | 19.75 |
| | StyleGAN2 [8] | 44.04 | 62.03 | 22.21 |
| Papa | BigGAN [7] | 31.37 | 53.45 | 24.99 |
| | ContraGAN [85] | 4.06 | 9.80 | 25.33 |
| | ReACGAN [10] | 26.75 | 50.17 | 21.74 |
| | StyleGAN2 [8] | 24.90 | 43.84 | 23.28 |
| Grandpa | BigGAN [7] | 23.68 | 46.56 | 24.40 |
| | ContraGAN [85] | 2.63 | 7.46 | 21.14 |
| | ReACGAN [10] | 24.48 | 50.56 | 18.65 |
| | StyleGAN2 [8] | 19.36 | 38.88 | 21.88 |
| ImageNet | BigGAN [7] | 37.28 | 61.96 | 20.57 |
| | ContraGAN [85] | 2.40 | 10.26 | 29.59 |
| | ReACGAN [10] | 23.14 | 50.31 | 15.65 |
| | StyleGAN2 [8] | 17.97 | 38.17 | 33.40 |

the evaluation using Inception Score (IS), Fréchet Inception Distance (FID), Precision & Recall, and Density & Coverage can be biased toward the pre-trained InceptionV3 model. Therefore, future studies should be conducted to identify which type of an evaluation backbone is appropriate for generative model evaluation.

IFID may lead to wrong conclusion. Intra-class FID (IFID) is the average of class conditional FIDs and is used to measure intra-class fidelity, diversity, and class conditioning performance in a scalar value. As Sajjadi *et al.* [124] stated, quantifying multiple characteristics into a scalar leads to confusing conclusions, and independently measuring each characteristic is required. For example, ContraGAN exhibits a lower FID value (21.14) than BigGAN (24.4) in the Grandpa-ImageNet generation experiment. However, the IFID value of ContraGAN (193.93) is exceptionally high compared to the IFID value of BigGAN (78.11). In Table 6, we notice that the exceptionally high IFID score of ContraGAN attributes to its limited capability in label conditioning, exemplifying that assessing generative models based solely on FID is not recommended.

Conditional generative models overlook class conditioning performance. While conditional GAN (cGAN) is devised to perform conditional image generation, studies that deal with cGAN put conditional image generation aside and mainly focus on generating realistic images. Although Classifier Accuracy Score (CAS) [146] is proposed to quantify class-conditional precision and recall of a generative model, it is not widely adopted due to the heavy computational cost induced by training ResNet [149] from scratch. We provide Top-1 and Top-5 classification accuracies using the InceptionV3 network instead. Table 6 indicates that ContraGAN models are poor at conditional image generation. In addition, BigGAN is better at class conditioning than ReACGAN, which differs from our expectation that classifier-based GANs would perform better in image conditioning.

Precision & Recall and Density & Coverage exhibit inconsistent results. Precision & Recall and Density & Coverage

TABLE 7. Benckmark table for GANs [7], [9], [10], [48], [63], autoregressive models [40], [41], [48], and diffusion probabilistic models [43], [44], [45], [46], [47]. We evaluate those generative models using TensorFlow-InceptionV3 [30] and the architecture-friendly resizer: PIL.BILINEAR for the postprocessing step. In the case of ImageNet-128 and ImageNet-256 experiments, we preprocess images using the best performing resizer among PIL.BILINEAR, PIL.BICUBIC, and PIL.LANCZOS based on Fréchet Distance. We mark \dagger if images are preprocessed using PIL.BILINEAR for evaluation, \ddagger for the case of PIL.BICUBIC, and $\ddagger\dagger$ for the case of PIL.LANCZOS. Inference time is computed using a single A100 GPU. Top-1 and Top-2 performances are indicated in red and blue, respectively.

| Model | # Param. | IS \uparrow | FID \downarrow | Precision \uparrow | Recall \uparrow | Density \uparrow | Coverage \uparrow | Inf. (s) | |
|-------------------------------|---|---------------|------------------|----------------------|-------------------|--------------------|---------------------|-------------|--------------|
| CIFAR10 | ReACGAN + DiffAug (Ours) [10] | 9.4M | 10.15 | 2.64 | 0.75 | 0.65 | 0.98 | 0.90 | 0.009 |
| | StyleGAN2-ADA [85] | 20.2M | 10.31 | 2.41 | 0.74 | 0.68 | 1.02 | 0.92 | 0.008 |
| | StyleGAN2-ADA (Ours) [85] | 20.2M | 10.53 | 2.31 | 0.75 | 0.69 | 1.04 | 0.93 | 0.008 |
| | StyleGAN2 + DiffAug + D2D-CE (Ours) [10] | 20.2M | 10.46 | 2.30 | 0.76 | 0.68 | 1.03 | 0.93 | 0.007 |
| | DDPM [43] | 35.2M | 9.73 | 3.23 | 0.78 | 0.67 | 1.10 | 0.93 | 15.422 |
| | DDPM++ [44] | 106.6M | 9.90 | 2.49 | 0.78 | 0.69 | 1.12 | 0.94 | 46.697 |
| | NCSN++ [44] | 107.6M | 10.08 | 2.27 | 0.77 | 0.70 | 1.07 | 0.94 | 99.304 |
| | LSGM [45] | - | 10.04 | 2.80 | 0.80 | 0.70 | 1.15 | 0.95 | - |
| | LSGM-ODE [45] | - | 10.07 | 2.09 | 0.77 | 0.71 | 1.03 | 0.94 | - |
| | CLD-SGM [47] | - | 9.88 | 2.38 | 0.78 | 0.69 | 1.12 | 0.94 | - |
| ImageNet-128 | StyleGAN-XL [63] | 18.0M | 11.03 | 1.88 | 0.77 | 0.59 | 1.08 | 0.94 | 0.010 |
| | BigGAN [7] $\ddagger\ddagger$ | 70M | 128.51 | 7.66 | 0.80 | 0.51 | 1.16 | 0.84 | 0.139 |
| | BigGAN (Ours) [7] \dagger | 70M | 98.51 | 8.54 | 0.75 | 0.60 | 0.96 | 0.82 | 0.015 |
| | BigGAN-Deep [7] $\ddagger\ddagger$ | 50M | 161.73 | 5.90 | 0.89 | 0.45 | 1.66 | 0.91 | 0.086 |
| | ReACGAN (Ours) [10] \dagger | 70M | 94.06 | 8.19 | 0.80 | 0.42 | 1.07 | 0.72 | 0.015 |
| | StyleGAN-XL [63] $\ddagger\ddagger\ddagger$ | 159M | 221.04 | 1.94 | 0.82 | 0.62 | 1.17 | 0.94 | 0.030 |
| ImageNet-256 | ADM-G [46] $\ddagger\ddagger$ | 464M | 156.22 | 3.05 | 0.81 | 0.68 | 1.13 | 0.95 | 23.513 |
| | BigGAN [7] \dagger | 164M | 185.52 | 7.75 | 0.83 | 0.47 | 1.34 | 0.85 | 0.324 |
| | BigGAN-Deep [7] \dagger | 112M | 224.46 | 6.95 | 0.89 | 0.38 | 1.67 | 0.88 | 0.177 |
| | VQGAN [48] \dagger | 1.5B | 314.61 | 5.20 | 0.81 | 0.57 | 1.03 | 0.85 | 6.551 |
| | StyleGAN-XL [63] $\ddagger\ddagger\ddagger$ | 166M | 297.62 | 2.32 | 0.82 | 0.61 | 1.16 | 0.93 | 0.040 |
| | ADM-G [46] \dagger | 608M | 207.86 | 4.48 | 0.84 | 0.62 | 1.19 | 0.94 | 50.574 |
| | ADM-G-U [46] \dagger | 673M | 240.24 | 4.01 | 0.85 | 0.62 | 1.22 | 0.95 | 49.609 |
| RQ-Transformer [41] \dagger | MaskGIT [40] \dagger | 227M | 216.38 | 5.40 | 0.87 | 0.60 | 1.37 | 0.94 | 5.313 |
| | RQ-Transformer [41] \dagger | 3.9B | 339.41 | 3.83 | 0.85 | 0.60 | 1.18 | 0.91 | 3.816 |

are pairs of metrics devised to disentangle the fidelity and diversity measurement from FID. However, we observe the Recall and Coverage metrics occasionally draw conflicting judgments, while precision and density metrics often reach a consensus. As shown in Table 5, InceptionV3-based Recall values of ReACGAN on CIFAR10, Baby/Papa/Grandpa-ImageNet, and ImageNet are 0.62, 0.43, 0.47, 0.41, and 0.38 respectively. Likewise, Recalls of BigGAN are 0.65, 0.47, 0.47, 0.48, and 0.65, respectively. These results indicate that generated images by ReACGAN generally have lower Recall values than BigGAN. Coverage values, however, imply that the generated images by ReACGAN hold better diversity than BigGAN. This phenomenon also holds for experiments using SwAV. As a result, we believe an extra study is required to accurately quantify the fidelity and diversity of generated images using those metrics.

8 OTHER APPROACHES

This chapter evaluates other popular generative models, such as GANs with excessive parameters, autoregressive models, and probabilistic diffusion models. These approaches require excessive computational resources to be trained from scratch, so we utilize pre-trained models provided by authors to get generated images. We follow our proposed evaluation protocol and preprocess the training images using the best-performed resizer among PIL.BILINEAR, PIL.BICUBIC, and PIL.LANCZOS based on FD value. The evaluation results are presented in Table 7.

StyleGAN-XL. StyleGAN-XL [63] tackles the limitation of the StyleGAN approaches that struggle with generating a diverse class of images, such as ImageNet. In the CIFAR10

generation experiment, StyleGAN-XL exhibits the FID of 1.88, being first to break the wall of 2. Despite the record, StyleGAN-XL shows the lowest recall value of 0.59 compared to the other competitive models. This phenomenon is unnatural since images with a low recall are expected to exhibit a high FID.

A similar phenomenon is observed in the ImageNet-256 experiment. While StyleGAN-XL achieves the lowest FID value compared to other models, ADM-G-U presents higher precision and recall scores than StyleGAN-XL. This contradicts our expectation that higher precision and recall scores should be correlated with a lower FID. We leave the analysis on this matter as a future work because a more careful and thorough analysis should take place to understand this corner case of evaluation metrics.

Auto-regressive (AR) and diffusion models. In Table 7, RQ-Transformer [41] shows the second best FID on ImageNet-256 followed by diffusion models ADM-G and ADM-G-U [46]. While StyleGAN-XL and RQ-Transformer exhibit lower FID values than ADM-G-U, we empirically confirm that ADM-G-U synthesizes more visually plausible images than StyleGAN-XL and RQ-Transformer. As can be seen in Figures A12, A13, and A14 in Appendix, the generated images from ADM-G-U appear to reflect the overall structure of objects more faithfully than StyleGAN-XL and RQ-Transformer. To our knowledge, ADM-G-U is the first model to generate reasonable facial images among ImageNet-trained generative models.

Comparing GANs with AR and diffusion models. Our experimental results suggest that AR and diffusion models show comparable or slightly worse FIDs with StyleGAN-XL (Table 7). However, the number of model parame-

StyleGAN-XL [63] on IMAGENET (256 × 256)



RQ-Transformer [41] on IMAGENET (256 × 256)



ADM-G-U [46] on IMAGENET (256 × 256)



Figure 6. Generated images by StyleGAN-XL (GAN) [63], RQ-Transformer (Auto Regressive Model) [41], and ADM-G-U (Denoising Diffusion Probabilistic Model) [46]. More qualitative results are displayed in Figures A12, A13, and A14.

ters in popular GAN models, such as BigGAN-Deep and StyleGAN-XL, is much smaller than AR and diffusion models. For instance, the number of parameters of BigGAN-Deep is 112M, but ADM-G and RQ-Transformer’s parameters are 608M and 3.9B, respectively. This implies that GANs are parameter-efficient and can be scaled up to overcome the poor structural coherency [64]. Furthermore, unlike AR and diffusion models, GANs do not require a consecutive inference process, enabling very fast sampling. For example, while it takes only 0.040 seconds to generate one sample with StyleGAN-XL, it takes upto 49.609 seconds with ADM-G-U (1000 DDPM). While there have been efforts to reduce denoising steps [150], [151], these models require a well-trained diffusion model for distillation. Moreover, the synthesis ability of the few-step diffusion models still lags behind the state-of-the-art GAN approach [64]. Hence, we emphasize that GANs are highly efficient regarding model size, inference speed, and synthesis ability.

9 FUTURE RESEARCH DIRECTION

Generative model evaluation. Our paper primarily focuses on the training and evaluation protocols, as well as providing a comprehensive benchmark. However, it is still very difficult to explicitly specify which metrics researchers should use to develop and evaluate generative models. While widely adopted metrics like IS and FID have served as guiding principles in visual generative AI, several cases were reported where they do not perfectly align with the human evaluation of realism [152]. As a result, human evaluation is a crucial component for the accurate evaluation of generative models. Recently, several attempts, including Zhou *et al.* [153]’s Human eYe Perceptual Evaluation (HYPE), try to introduce human-in-the-loop evaluation based on psychophysics. However, the time and cost involved in human evaluations limit the active use of this method in the community, and as a consequence,

only a few selected models like SNGAN [107], BigGAN [7], and StyleGAN [54] offer human evaluation results. Given these circumstances, we believe developing more reliable human/automatic procedures will be critical in the future.

Image synthesis on open-world visual images. Recently, due to the amazing results from text-to-image synthesis models through a diffusion process, the generative modeling community is experiencing a rapid shift in technological trends from GANs to diffusion or auto-regressive (AR) models. With the development of text-to-image diffusion models, such as GLIDE [154], DALL-E-2 [92], and Imagen [93], an enormous amount of GPUs are being devoted to training diffusion or AR models. GANs have yet to be actively utilized for text-to-image synthesis for their brittle nature of adversarial dynamics and their inability to synthesize structural details of objects within images faithfully. However, recent models, such as GigaGAN [64] and StyleGAN-T [65] have demonstrated that GANs can be also scaled up and used for text-to-image synthesis on open-world datasets, such as LAION2B-en [155] and COYO-700M [156]. This reveals that GANs can potentially serve as a breakthrough alternative to existing text-to-image models, which have suffered from slow inference time.

On the other hand, there exist efforts to reduce the inference time of diffusion models through techniques like consistency distillation [151] and progressive distillation [150], [157]. These approaches aim to make the synthesis process more efficient. We anticipate that these single-step generative models, including GANs, will be applied to various real-world applications. For example, one-step diffusion models can be effectively employed for customization tasks [158] that involve fine-tuning the model, while GANs can be utilized for image manipulation tasks [159] by leveraging the linear latent space property.

10 CONCLUSION

In this paper, we have introduced improper practices that arise when training and evaluating GAN. By correcting the above practices, we have presented training and evaluation protocols focusing on high-quality image generation and fair evaluation. Since current image synthesis benchmarks are not constructed using the proposed protocols, we have implemented extensive types of GANs based on our proposed GAN taxonomy to train and evaluate GANs from scratch. We provide StudioGAN, a software library of various GAN and relevant modules. Using the training and evaluation protocols and StudioGAN, we have offered a large-scale GAN evaluation benchmark and explained intriguing findings and potential hazards, which cast important questions for generative model development. In addition, we have evaluated recent image generation approaches such as ADM and RQ-Transformer. We hope StudioGAN serves as a cornerstone to explore GAN approaches and boosts the development of a next-generation generative AI.

ACKNOWLEDGMENTS

This work is supported by IITP grants 2019-0-01906 (POSTECH AI Graduate School Program), 2021-0-00537 (Image Restoration), and 2022-0-00290 (Visual Intelligence) funded by the Korean government (MSIT).

REFERENCES

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.
- [2] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv preprint arXiv:1511.06434*, 2016.
- [3] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016.
- [4] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled Generative Adversarial Networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [5] J. J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based Generative Adversarial Network," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [6] M. Lin, "Softmax gan," *arXiv preprint arXiv:1704.06191*, 2017.
- [7] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," in *Proceedings of the International Conference on Learning Representations*, 2019.
- [8] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [9] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [10] M. Kang, W. Shim, M. Cho, and J. Park, "Rebooting acgan: Auxiliary classifier gans with stable training," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [11] N. Kumari, R. Zhang, E. Shechtman, and J.-Y. Zhu, "Ensembling off-the-shelf models for gan training," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.
- [13] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [14] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [15] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, "Few-Shot Unsupervised Image-to-Image Translation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.
- [16] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive Learning for Unpaired Image-to-Image Translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [17] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or, "Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [18] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2017.
- [19] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018.
- [20] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.
- [21] J. Guan, C. Pan, S. Li, and D. Yu, "Srdgan: learning the noise prior for super resolution with dual generative adversarial networks," *arXiv preprint arXiv:1903.11821*, 2019.
- [22] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [23] E. C. et. al. and Marco Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [24] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [25] J. Gu, L. Liu, P. Wang, and C. Theobalt, "StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image Synthesis," *arXiv preprint arXiv:2110.08985*, 2021.
- [26] P. Zhou, L. Xie, B. Ni, and Q. Tian, "CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis," *arXiv preprint arXiv:2110.09788*, 2021.
- [27] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved Techniques for Training GANs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.
- [28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [29] G. Parmar, R. Zhang, and J.-Y. Zhu, "On Buggy Resizing Libraries and Surprising Subtleties in FID Calculation," *arXiv preprint arXiv:2104.11222*, 2021.
- [30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [31] T. Kynkänniemi, T. Karras, M. Aittala, T. Aila, and J. Lehtinen, "The Role of ImageNet Classes in Fréchet Inception Distance," *arXiv preprint arXiv:2203.06026*, 2022.
- [32] T. Kynkänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila, "Improved Precision and Recall Metric for Assessing Generative

- Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [33] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable Fidelity and Diversity Metrics for Generative Models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.
- [34] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [35] S. Morozov, A. Voynov, and A. Babenko, "On self-supervised image representations for gan evaluation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- [36] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging Properties in Self-Supervised Vision Transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [37] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [38] H. Zhang, Z. Zhang, A. Odena, and H. Lee, "Consistency Regularization for Generative Adversarial Networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.
- [39] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, and B. Guo, "Vector Quantized Diffusion Model for Text-to-Image Synthesis," *arXiv preprint arXiv:2111.14822*, 2021.
- [40] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman, "MaskGIT: Masked Generative Image Transformer," *arXiv preprint arXiv:2202.04200*, 2022.
- [41] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han, "Autoregressive Image Generation using Residual Quantization," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [42] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [43] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [44] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- [45] A. Vahdat, K. Kreis, and J. Kautz, "Score-based generative modeling in latent space," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [46] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [47] T. Dockhorn, A. Vahdat, and K. Kreis, "Score-Based Generative Modeling with Critically-Damped Langevin Diffusion," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- [48] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [49] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.
- [50] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.
- [51] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "CVAE-GAN: fine-grained image generation through asymmetric training," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.
- [52] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [53] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," *arXiv preprint arXiv:1710.10196*, 2018.
- [54] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [55] A. Karnewar and O. Wang, "Msg-gan: Multi-scale gradients for generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [56] B. Liu, Y. Zhu, K. Song, and A. Elgammal, "Towards faster and stabilized gan training for high-fidelity few-shot image synthesis," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.
- [57] Y. Jiang, S. Chang, and Z. Wang, "Transgan: Two pure transformers can make one strong gan, and that can scale up," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [58] L. Zhao, Z. Zhang, T. Chen, D. Metaxas, and H. Zhang, "Improved transformer for high-resolution gans," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [59] D. A. Hudson and L. Zitnick, "Generative adversarial transformers," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.
- [60] A. Sauer, K. Chitta, J. Müller, and A. Geiger, "Projected GANs Converge Faster," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [61] K. Lee, H. Chang, L. Jiang, H. Zhang, Z. Tu, and C. Liu, "ViT-GAN: Training GANs with vision transformers," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- [62] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang, and B. Guo, "Styleswin: Transformer-based gan for high-resolution image generation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [63] A. Sauer, K. Schwarz, and A. Geiger, "StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets," *arXiv preprint arXiv:2202.00273*, 2022.
- [64] M. Kang, J.-Y. Zhu, R. Zhang, J. Park, E. Shechtman, S. Paris, and T. Park, "Scaling up GANs for Text-to-Image Synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [65] A. Sauer, T. Karras, S. Laine, A. Geiger, and T. Aila, "StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis," *arXiv.org*, vol. abs/2301.09515, 2023.
- [66] T. Z. P. Isola, J. Zhu and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [67] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.
- [68] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [69] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.
- [70] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [71] S. Mo, M. Cho, and J. Shin, "InstaGAN: Instance-aware Image-to-Image Translation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [72] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [73] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, "Few-shot unsupervised image-to-image translation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.
- [74] J. Kim, M. Kim, H. Kang, and K. H. Lee, "U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.
- [75] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "StarGAN v2: Diverse Image Synthesis for Multiple Domains," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

- [76] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [77] A. Casanova, M. Careil, J. Verbeek, M. Drozdzal, and A. Romero-Soriano, "Instance-Conditioned GAN," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [78] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv preprint arXiv 1411.1784*, 2014.
- [79] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis with Auxiliary Classifier GANs," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.
- [80] Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang, J. Wang, and Y. Yu, "Activation Maximization Generative Adversarial Nets," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [81] T. Miyato and M. Koyama, "cGANs with Projection Discriminator," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [82] A. Siarohin, E. Sangineto, and N. Sebe, "Whitening and Coloring Batch Transform for GANs," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [83] M. Gong, Y. Xu, C. Li, K. Zhang, and K. Batmanghelich, "Twin Auxiliary Classifiers GAN," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [84] I. Kavalerov, W. Czaja, and R. Chellappa, "A multi-class hinge loss for conditional gans," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.
- [85] M. Kang and J. Park, "ContraGAN: Contrastive Learning for Conditional Image Generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [86] P. Zhou, L. Xie, B. Ni, and Q. Tian, "Omni-GAN: On the Secrets of cGANs and Beyond," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [87] L. Hou, Q. Cao, H. Shen, and X. Cheng, "cGANs with Auxiliary Discriminative Classifier," *arXiv preprint arXiv:2107.10060*, 2021.
- [88] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [89] M. Zhu, P. Pan, W. Chen, and Y. Yang, "Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [90] H. Zhang, J. Y. Koh, J. Baldridge, H. Lee, and Y. Yang, "Cross-modal contrastive learning for text-to-image generation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [91] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.
- [92] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," *arXiv preprint arXiv:2204.06125*, 2022.
- [93] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes *et al.*, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," *arXiv preprint arXiv:2205.11487*, 2022.
- [94] V. Dumoulin, J. Shlens, and M. Kudlur, "A Learned Representation For Artistic Style," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [95] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [96] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv 1701.07875*, 2017.
- [97] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [98] N. Kodali, J. Hays, J. D. Abernethy, and Z. Kira, "On Convergence and Stability of GANs," *arXiv preprint arXiv 1705.07215*, 2018.
- [99] H. Petzka, A. Fischer, and D. Lukovicov, "On the regularization of wasserstein gans," *arXiv preprint arXiv:1709.08894*, 2017.
- [100] A. Müller, "Integral probability metrics and their generating classes of functions," *Advances in Applied Probability*, 1997.
- [101] Y. Mroueh, T. Sercu, and V. Goel, "McGan: mean and covariance feature matching GAN," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.
- [102] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "MMD GAN: Towards Deeper Understanding of Moment Matching Network," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [103] Y. Mroueh and T. Sercu, "Fisher gan," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [104] M. G. e. a. Bellemare, "The cramer distance as a solution to biased wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017.
- [105] S. W. Park and J. Kwon, "SphereGAN: Sphere generative adversarial network based on geometric moment matching and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [106] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [107] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [108] L. Mescheder, S. Nowozin, and A. Geiger, "Which Training Methods for GANs do actually Converge?" in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.
- [109] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han, "Differentiable augmentation for data-efficient gan training," *arXiv preprint arXiv 2006.10738*, 2020.
- [110] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training Generative Adversarial Networks with Limited Data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [111] L. Jiang, B. Dai, W. Wu, and C. C. Loy, "Deceive D: Adaptive Pseudo Augmentation for GAN Training with Limited Data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [112] H.-Y. Tseng, L. Jiang, C. Liu, M.-H. Yang, and W. Yang, "Regularizing Generative Adversarial Networks Under Limited Data," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [113] J. et al., "Tiny ImageNet Visual Recognition Challenge," <https://tiny-imagenet.herokuapp.com>.
- [114] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1981.
- [115] K. Turkowski, "Filters for common resampling tasks," *Graphics gems*, 1990.
- [116] M. A. et. al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015.
- [117] A. e. a. Paszke, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [118] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [119] P. Umesh, "Image Processing in Python," *CSI Communications*, 2012.
- [120] A. Brock, "BigGAN-PyTorch," <https://github.com/ajbrock/BigGAN-PyTorch>, 2019.
- [121] C. R. H. et. al., "Array programming with NumPy," *Nature*, 2020.
- [122] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Ph.D. dissertation, University of Toronto, 2012.
- [123] M. Birikowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [124] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- [125] S. Liu, Y. Wei, J. Lu, and J. Zhou, "An improved evaluation framework for generative adversarial networks," *arXiv preprint arXiv:1803.07474*, 2018.
- [126] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.
- [127] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture

- for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.
- [128] Q. Li, "Training results(IS and FID) are not good as yours with same training process," <https://github.com/ajbrock/BigGAN-PyTorch/issues/23>, 2019.
- [129] Y. Zhao, C. Li, P. Yu, J. Gao, and C. Chen, "Feature Quantization Improves GAN Training," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.
- [130] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh et al., "Mixed Precision Training," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [131] openmmlab, "MMGeneration: Openmmlab generative model toolbox and benchmark," <https://github.com/open-mmlab/mmgeneration>, 2021.
- [132] K. S. Lee and C. Town, "Mimicry: Towards the Reproducibility of GAN Research," in *CVPRW on AI for Content Creation*, 2020.
- [133] Y. Wu, J. Donahue, D. Balduzzi, K. Simonyan, and T. P. Lillicrap, "LOGAN: Latent Optimisation for Generative Adversarial Networks," *arXiv preprint arXiv 1912.00953*, 2019.
- [134] Z. Zhao, S. Singh, H. Lee, Z. Zhang, A. Odena, and H. Zhang, "Improved Consistency Regularization for GANs," *arXiv preprint arXiv 2002.04724*, 2020.
- [135] W. Shim and M. Cho, "CircleGAN: Generative Adversarial Learning across Spherical Circles," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [136] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," <https://github.com/brain-research/self-attention-gan>, 2018.
- [137] J. Mao, "Synchronized-BatchNorm-PyTorch Public," <https://github.com/vacancy/Synchronized-BatchNorm-PyTorch>, 2018.
- [138] openai, "improved-gan," <https://github.com/openai/improved-gan>, 2016.
- [139] mseitzer, "FID score for PyTorch," <https://github.com/mseitzer/pytorch-fid>, 2018.
- [140] clovaai, "Reliable Fidelity and Diversity Metrics for Generative Models," <https://github.com/clovaai/generative-evaluation-prdc>, 2020.
- [141] facebookresearch, "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments," <https://github.com/facebookresearch/swav>, 2020.
- [142] microsoft, "Swin Transformer," <https://github.com/microsoft/Swin-Transformer>, 2021.
- [143] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, 2004.
- [144] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [145] S. Gu, J. Bao, D. Chen, and F. Wen, "Giga: Generated image quality assessment," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [146] S. V. Ravuri and O. Vinyals, "Classification Accuracy Score for Conditional Generative Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [147] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least Squares Generative Adversarial Networks," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.
- [148] J. H. Lim and J. C. Ye, "Geometric GAN," *arXiv preprint arXiv 1705.02894*, 2017.
- [149] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [150] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans, "On distillation of guided diffusion models," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [151] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.
- [152] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding*, 2019.
- [153] S. Zhou, M. Gordon, R. Krishna, A. Narcomey, L. F. Fei-Fei, and M. Bernstein, "Hype: A benchmark for human eye perceptual evaluation of generative models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [154] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.
- [155] C. e. a. Schuhmann, "LAION-5B: An open large-scale dataset for training next generation image-text models," *arXiv preprint arXiv:2210.08402*, 2022.
- [156] M. Byeon, B. Park, H. Kim, S. Lee, W. Baek, and S. Kim, "COYO-700M: Image-Text Pair Dataset," <https://github.com/kakaobrain/coyo-dataset>, 2022.
- [157] T. Salimans and J. Ho, "Progressive Distillation for Fast Sampling of Diffusion Models," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.
- [158] N. Kumari, B. Zhang, R. Zhang, E. Shechtman, and J.-Y. Zhu, "Multi-concept customization of text-to-image diffusion," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [159] X. Pan, A. Tewari, T. Leimkühler, L. Liu, A. Meka, and C. Theobalt, "Drag your gan: Interactive point-based manipulation on the generative image manifold," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023.
- [160] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.
- [161] Z. Zhou, J. Liang, Y. Song, L. Yu, H. Wang, W. Zhang, Y. Yu, and Z. Zhang, "Lipschitz generative adversarial nets," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.
- [162] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Neural Photo Editing with Introspective Adversarial Networks," in *in Proceedings of the International Conference on Learning Representation*, 2017.
- [163] S. Sinha, Z. Zhao, A. Goyal, C. Raffel, and A. Odena, "Top-k Training of GANs: Improving GAN Performance by Throwing Away Bad Samples," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.



Minguk Kang received the BS degree from Pusan National University, South Korea, in 2019. He is currently working toward the PhD from the Graduate School of Artificial Intelligence, POSTECH, South Korea, where he created the StudioGAN library. His research interests include generative modeling, with applications in computer vision. He has authored three academic papers in NeurIPS and CVPR.



Joonghyuk Shin is a graduate student at the Department of Computer Science and Engineering, Seoul National University, South Korea. He received BS degree from the Department of Computer Science and Engineering, POSTECH in 2023. Along with Minguk, he is one of the core contributors of the StudioGAN library. His research interests include generative modeling and its diverse applications in computer vision.



Jaesik Park is an Assistant Professor at the Department of Computer Science and Engineering, Seoul National University. He received his Bachelor's degree from Hanyang University (2009), and he received his Master's degree (2011) and Ph.D. degree (2015) from KAIST. He worked at Intel as a research scientist (2015-2019), and he was a faculty member at POSTECH (2019-2023). His research interests include image synthesis and 3D scene understanding.

APPENDIX

A1 SUPPORTED MODULES IN STUDIOGAN

GANs:

- DCGAN [2]
- InfoGAN [160]
- LSGAN [147]
- Geometric GAN [148]
- WGAN [96]
- WGAN-GP [52]
- WGAN-DRA [98]
- ACGAN [79]
- PD-GAN [81]
- SNGAN [107]
- SAGAN [49]
- TACGAN [83]
- LGAN [161]
- Unconditional BigGAN [7]
- BigGAN [7]
- BigGAN-Deep [7]
- StyleGAN2 [8]
- CRGAN [38]
- ICRCGAN [134]
- LOGAN [133]
- ContraGAN [85]
- MHGAN [84]
- BigGAN-DiffAug [109]
- StyleGAN2-ADA [110]
- BigGAN-LeCam [112], ADCGAN [87]
- ReACGAN [10]
- StyleGAN2-APA [111]
- StyleGAN3-t [9]
- StyleGAN3-r [9]

Network Architectures:

- DCGAN [2]
- ResNetGAN [52]
- SAGAN [49]
- BigGAN [7]
- BigGAN-Deep [7]
- StyleGAN2 [8]
- StyleGAN3 [9].

Conditioning methods:

- cGAN [78]
- ACGAN [79]
- cBN [81], [94], [95]
- PD-GAN [81]
- TACGAN [83]
- MHGAN [84]
- ContraGAN [85]
- ADCGAN [87]
- ReACGAN [10]

Adversarial losses:

- WGAN [96]
- Vanilla GAN [1]
- LSGAN [147]
- Geometric GAN [148]

Regularizations:

- InfoGAN [160]

- WGAN-GP [52]
- WGAN-DRA [98]
- R1 Regularization [108]
- SNGAN [107]
- Orthogonal Regularization [162]
- PPL Regularization [8]
- LGAN [161]
- CRGAN [38]
- LOGAN [133]
- Top-k Training [163]
- ICRCGAN [134]
- LeCam [112]

Differentiable Augmentations:

- DiffAug [109]
- ADA [110]
- APA [111]

A2 DATASETS

CIFAR10 [122] is comprised of 50k train, and 10k test RGB images of 32×32 resolution. There are a total of ten classes, resulting in 5k train images and 1k test images per class. For the CIFAR10 10% dataset, we randomly chose 500 images for each class. And for the CIFAR10 30% dataset, we added 1000 more randomly selected images on top of the 10% dataset.

Baby, Papa, and Grandpa ImageNet are newly introduced subsets of the ImageNet dataset to replace old Tiny-ImageNet [113]. Using InceptionV3 network's top-1 classification accuracy of ImageNet 1,000 classes, we sample 1~100 (most accurate), 451~550 (modest), and 901~1,000 (difficult) classes and group them into Baby, Papa, and Grandpa ImageNet, respectively. With Baby ImageNet being the easiest and Grandpa ImageNet being the hardest, images are center cropped and then resized into 64×64 resolution using the high-quality PIL.LANCZOS resizer [115] explained in section 4.1. The resulting images show much less degradation and aliasing artifacts compared to the legacy Tiny-ImageNet as can be seen in Figure A1.

ImageNet [106] provides around 1.2M and 50k RGB images in 1,000 classes for training and validation. We apply center crop to every image and resize images to 128×128 pixels using PIL.LANCZOS resizer as stated above.

AFHQv2 [75] is an improved version of the previous Animal Face High-Quality dataset (AFHQ) [75] by utilizing Lanczos resampling instead of nearest neighbor downsampling. Total 15,803 RGB images (14,336 train, 1,467 test) with 512×512 resolution fall into three classes (cat, dog, and wild). Likewise, we resize images to 512×512 pixels using Lanczos resizer for Tables 3, 5, A3, A4 and 256×256 pixels for Table A1 whose purpose is identifying the effectiveness of modules for data-efficient training.

FFHQ (FQ for short) [54] consists of 70k high-quality face images at 1024×1024 resolution. The dataset contains abundant facial variations including gender, age, ethnicity, and image background.



Figure A1. Images sampled from Tiny-ImageNet dataset (64×64 pix.) [113] (Shown on the left) and from Baby-ImageNet dataset (64×64 pix.) (Shown on the right). The images of Tiny-ImageNet exhibit severe image degradation due to improper data preprocessing. Images from Baby-ImageNet are resized to 64×64 resolution using PIL.LANCZOS resizer, which results in better image quality.

A3 TRAINING AND EVALUATION SETUPS

For experiments, we report averaged value of three runs for CIFAR10, two runs for Baby/Papa/Grandpa-ImageNet, and a single run for ImageNet, AFHQv2, and FFHQ. All experiments were performed using mixed-precision training, and experiments that had collapsed at the early stage were re-executed. We empirically find that unexpected training failures occasionally occur when mixed precision is applied. We attribute this phenomenon to the extremely sensitive GAN dynamics. Using mixed precision requires scaling of floating points between FP16 and FP32, and some information loss is inevitable. While this may not be a big problem for most neural networks, we suspect that this has enormous effects on GAN dynamics which are susceptible to even the slightest gradient changes.

With a lone exception of the CIFAR10 experiment that was done by a single GPU, all models were trained with 4-GPUs with proper accumulation for batch statistics. Since the variation during the evaluation process is relatively small, we evaluate every model once. (*i.e.* One evaluation for each of the three runs for CIFAR10, summing up to three for each configuration) All evaluation was done on a single GPU.

Further details for training and evaluation are summarized below:

- We use PIL.LANCZOS resizer to preprocess training images (Sec. 4.1) and apply a random horizontal flip.
- We use PIL.BILINEAR resizer to process real and fake images for evaluation if InceptionV3 and SwAV backbones are applied. For the evaluation using Swin-T, we use PIL.BICUBIC resizer (Sec. 4.2).
- For every experiment, we use the training set for the evaluation, not the validation set.
- We use the same number of generated images as the training images for Fréchet Distance (FD), Precision, Recall, Density, and Coverage calculation. For the experiments using Baby/Papa/Grandpa-ImageNet and Ima-

geNet, we exceptionally use 50k fake images against a complete training set as real images.

- When evaluating GANs, we always apply the standing statistics technique [7] except for the experiments using StyleGAN2 [8] or StyleGAN3 [9]. As mentioned in Sec. 4.5, two hyperparameters (*maximum batch size* and *number of repetition*) are required for applying standing statistics. Empirically, we found that these hyperparameters perform nicely, when set to the same value of training batch size. (*e.g.* If batch size was 256, we use 256 for both *maximum batch size* and *number of repetition*.)
- We use PyTorch implementation of InceptionV3 [139] whose network weights are converted from OpenAI’s TensorFlow-InceptionV3 network [138]. In the SwAV and Swin-T models, we use authors’ official codes and pretrained network weights [141], [142].
- When we calculate Intra-Class FD, we use the training images selected for each class as the reference and utilize the same amount of generated images for comparison.

A4 EVALUATING DATA-EFFICIENT TECHNIQUES

Experiment setup. We test various data-efficient training techniques to identify which technique works well in data-hungry situations. We select augmentation-based methods (DiffAug [109], ADA [110], and APA [111]) and regularization-based method (LeCam [112]). We also test whether an augmentation-based method can harmonize with the regularization-based method. We test DiffAug and LeCam with the same hyperparameters used in DiffAug and LeCam experiments. Figure A2 shows results for 10%, 30%, and 100% of CIFAR10 dataset on ResNet, BigGAN, and StyleGAN2 backbones, and Table A1 shows results for AFHQv2 dataset in 256×256 resolution with BigGAN and StyleGAN2 backbones. We train CIFAR10 models for 500k iterations or fewer in case of early collapse for a fair evaluation. Note that the FID value of the StyleGAN2-ADA

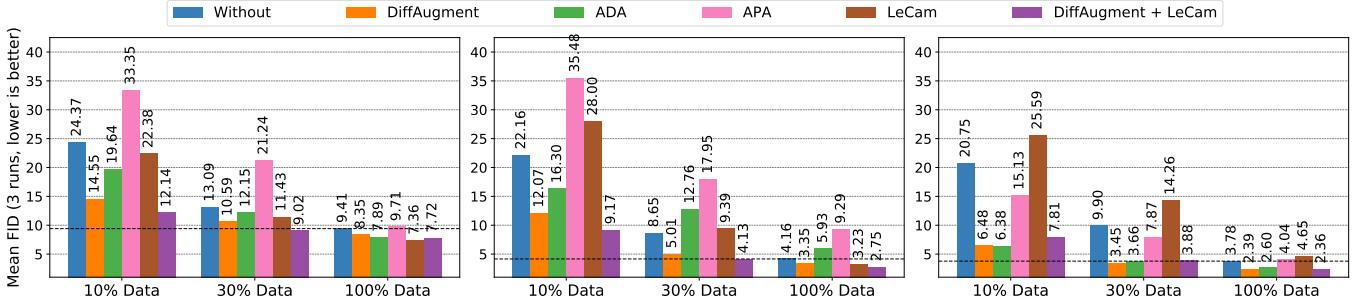


Figure A2. FID values of various data-efficient training methods applied to three GAN backbones: ResNet [52], BigGAN [7], and StyleGAN2 [8]. The dotted line indicates the performance of GAN trained using the full images without any data-efficient training trick.

TABLE A1. Data-efficient training results. We resize AFHQv2 to 256×256 resolutions using PIL.LANCZOS resizer for training. Top-1 and Top-2 performances are indicated in red and blue, respectively.

| Model + Data-efficient trick | IS \uparrow | FID \downarrow | Precision \uparrow | Recall \uparrow | Density \uparrow | Coverage \uparrow | Avg. Rank | |
|------------------------------|--|------------------|----------------------|-------------------|--------------------|---------------------|-----------|-------|
| AFHQv2 | BigGAN [7] | 7.64 | 29.13 | 0.86 | 0.02 | 1.30 | 0.45 | 8.00 |
| | BigGAN + DiffAug [109] | 6.81 | 51.58 | 0.78 | 0.00 | 1.16 | 0.20 | 10.00 |
| | BigGAN + ADA [110] | 11.70 | 5.52 | 0.84 | 0.51 | 1.32 | 0.85 | 4.33 |
| | BigGAN + LeCam [112] | 8.27 | 30.72 | 0.89 | 0.05 | 1.49 | 0.46 | 6.17 |
| | BigGAN + APA [111] | 6.20 | 68.37 | 0.31 | 0.00 | 0.17 | 0.05 | 11.67 |
| | BigGAN + DiffAug + LeCam [109], [112] | 6.31 | 54.03 | 0.42 | 0.00 | 0.24 | 0.07 | 10.83 |
| AFHQv2 | StyleGAN2 [8] | 11.68 | 7.21 | 0.87 | 0.36 | 1.45 | 0.82 | 5.67 |
| | StyleGAN2 + DiffAug [109] | 12.29 | 6.13 | 0.89 | 0.31 | 1.57 | 0.84 | 3.17 |
| | StyleGAN2 + ADA [110] | 12.31 | 5.13 | 0.86 | 0.48 | 1.33 | 0.85 | 2.83 |
| | StyleGAN2 + LeCam [112] | 11.88 | 7.16 | 0.86 | 0.37 | 1.43 | 0.82 | 5.17 |
| | StyleGAN2 + APA [111] | 11.7 | 5.44 | 0.83 | 0.53 | 1.18 | 0.83 | 5.17 |
| | StyleGAN2 + DiffAug + LeCam [109], [112] | 11.98 | 6.05 | 0.88 | 0.41 | 1.53 | 0.84 | 3.17 |

model continues to drop until 2.31 at around 1.5M iterations, as can be seen in Table 7, but here we train all models up to 500k iterations to check the tendency. All AFHQv2 models were trained for 200k iterations, and it is reported in Table 7.

Results. The results reveal that as the amount of data decreases, the FID scores notably worsen. However, figure A2 provides evidence that utilizing appropriate data-efficient training techniques can significantly alleviate the adverse effects of training GAN with limited data, leading to improved results. DiffAug and DiffAug + LeCam work best in generating CIFAR10 images in the extreme case when only 10% or 30% is provided. BigGAN with ADA occasionally exhibits worse performances than those without data-efficient techniques. However, StyleGAN2 consistently improves with ADA. Furthermore, Table A1 shows that BigGAN can only successfully generate AFHQv2 images when trained with ADA, implying that choosing an appropriate data-efficient training method is essential in limited data scenarios, and ADA is often a favorable option.

Figure A2 shows that DiffAug with LeCam presents the best or second best FID values across three GAN backbones on CIFAR10 except for StyleGAN2 with 10% and 30% of data. It's also worthwhile to note that BigGAN with DiffAug + LeCam gives a decent FID of 4.13 with only 30% of the data, which is quite impressive considering that BigGAN without any techniques displays FID of 4.16 with complete CIFAR10 data. While the combination of DiffAug and LeCam does not present better performance than ADA in the AFHQv2 experiment, we think that DiffAug can work in

harmony with LeCam when appropriate hyperparameters for GAN training are selected.

We also observe APA and LeCam sometimes showing substandard performance when applied solely. In the case of APA, we think it is because APA was mainly targeted for the StyleGAN backbone concentrating on center-aligned datasets like FFHQ and AFHQv2. Figure A2(c) and Table A1 prove that APA works better with StyleGAN2 backbone and even better when combined with center aligned images like AFHQv2. LeCam, on the other hand, was originally devised for diversified class conditional datasets like CIFAR10 and ImageNet with considerations for both BigGAN and StyleGAN2 backbones. Authors of LeCam emphasize the power of this regularization when combined with augmentation methods. Our experiment results, although not perfectly fit, mostly correspond to the authors' idea that the combination of DiffAug + LeCam shows the best results for CIFAR10 across most settings. Although original paper [112] suggests LeCam, alone works better than DiffAug on 10% ImageNet, our experiment results on 10% and 30% CIFAR10 display the limitation of this regularization-based method when applied solely in hugely data-hungry circumstances.

AFHQv2 generation experiments in Table A1 show that StyleGAN2 + ADA presents better Recall and Coverage values than that of StyleGAN2 + DiffAug. On the other hand, StyleGAN2 + ADA presents lower Precision and Density values than that of StyleGAN2 + DiffAug. DiffAug applies three types of image augmentations with fixed probability, whereas ADA applies diverse augmentations that are de-

TABLE A2. Impact of post resizer. We evaluate various generative models trained on ImageNet-256 with varying combination of backbone network and post resizer.

| | BigGAN [7] | BigGAN-Deep [7] | VQGAN [48] | StyleGAN-XL [63] | ADM-G-U [46] | ADM-G [46] | MaskGIT [40] | RQ-Transformer [41] |
|---------|---------------|-----------------|------------|------------------|--------------|------------|--------------|---------------------|
| Swin-T | Bilinear | 7.75 | 6.95 | 5.20 | 2.51 | 4.01 | 4.48 | 5.40 |
| | Bicubic [114] | 8.11 | 7.26 | 5.27 | 2.34 | 4.10 | 4.94 | 5.74 |
| | Lanczos [115] | 8.26 | 7.41 | 5.33 | 2.32 | 4.16 | 5.16 | 5.92 |
| SwAV | Bilinear | 4.02 | 3.57 | 3.03 | 1.10 | 1.78 | 1.54 | 2.48 |
| | Bicubic [114] | 4.02 | 3.60 | 3.03 | 1.08 | 1.81 | 1.60 | 2.49 |
| | Lanczos [115] | 4.05 | 3.63 | 3.05 | 1.08 | 1.83 | 1.62 | 2.52 |
| IncepV3 | Bilinear | 19.15 | 13.93 | 13.45 | 8.61 | 9.61 | 8.82 | 13.33 |
| | Bicubic [114] | 23.59 | 16.97 | 11.66 | 5.43 | 7.81 | 8.37 | 16.88 |
| | Lanczos [115] | 30.36 | 22.5 | 13.79 | 6.18 | 9.76 | 11.16 | 22.61 |

terminated probabilistically. We speculate that such nature of DiffAug makes images more discriminative but less diverse than ADA.

A5 IMPACT OF POST-RESIZER

Table 7 presents Fréchet Distances for models trained on ImageNet-256 with varying combinations of the backbone network and post-resizer. While SwAV and InceptionV3 backbones are the most and second robust to the post-resizer change, we observe that Swin-T backbone evaluations are heavily affected by different post-resizing methods. For Swin-T backbone evaluation results, rankings among other models frequently change when a different post-resizer is adopted. VQGAN, StyleGAN-XL, and ADM-G-U notably perform better when choosing a bicubic resizer. Furthermore, if models are evaluated without a consistent post-resizer, we even observe BigGAN with bilinear resizer (19.15) outperforming BigGAN-Deep with Lanczos resizer (30.36) in the Swin-T backbone. For the InceptionV3 backbone, the bilinear resizer usually shows the lowest score with a noticeable gap with the Lanczos resizer. As a result, we summarize that model rankings can change not only by the different backbone but also by different post-resizers, adding more significance to a unified evaluation protocol with a consistent post-resizer.

A6 EVALUATION USING OTHER BACKBONES

We provide evaluation results using SwAV [34] and Swin-T [37] backbones in Tables A3 and A4. Using the quantitative results, we visualize the results in the form of scatter plot in Figures A5, A7, A6, A8, A10, and A11.

TABLE A3. Benchmark table evaluated using **PyTorch-SwAV** [34]. The resolutions of CIFAR10 [122], Baby/Papa/Grandpa ImageNet, ImageNet [106], AFHQv2 [9], [75], and FQ [54] datasets are 32, 64, 128, 512, and 1024, respectively. Top-1 and Top-2 performances are indicated in red and blue, respectively.

| | PyTorch-SwAV [34] | SS \uparrow | FSD \downarrow | S-Precision \uparrow | S-Recall \uparrow | S-Density \uparrow | S-Coverage \uparrow | IFSD \downarrow |
|------------------|---|---|---|--|--|--|--|---|
| CIFAR10 | DCGAN [1] LSGAN [147] Geometric GAN [148] | 6.32 7.05 6.65 | 7.28 6.64 6.67 | 0.77 0.72 0.77 | 0.00 0.03 0.00 | 0.81 0.71 0.86 | 0.28 0.35 0.31 | 13.13 12.81 12.77 |
| | ACGAN-Mod [79] WGAN [96] DRAGAN [98] WGAN-GP [52] PD-GAN [81] SNGAN [107] SAGAN [49] TACGAN [83] LOGAN [133] | 7.46 4.21 7.10 5.78 8.38 10.79 10.26 8.18 8.61 | 5.56 21.02 5.07 9.87 5.06 1.39 0.86 4.64 2.64 | 0.78 0.17 0.81 0.68 0.77 0.36 0.35 0.75 0.85 | 0.00 0.00 0.06 0.05 0.01 0.27 | 0.87 0.05 1.03 0.62 0.85 1.23 1.14 0.81 1.14 | 0.33 0.02 0.44 0.28 0.39 0.79 0.77 0.40 0.65 | 9.64 25.21 11.28 15.35 8.24 2.61 2.85 8.10 9.45 |
| Baby-ImageNet | BigGAN [7] CRGAN [38] MHGAN [84] ICRGAN [134] ContraGAN [85] BigGAN + DiffAug [109] BigGAN + LeCam [112] ADCGAN [87] ReACGAN [10] | 13.63 14.21 13.61 14.00 13.27 13.46 13.98 13.35 13.44 | 0.60 0.46 0.57 0.53 0.78 0.56 0.45 0.53 0.56 | 0.85 0.84 0.85 0.86 0.85 0.86 0.84 0.85 0.87 | 0.54 0.56 0.51 0.50 0.50 0.55 0.58 0.56 0.46 | 1.11 1.06 1.10 1.15 1.09 1.14 1.07 1.10 1.28 | 0.89 0.90 0.89 0.91 0.86 0.90 0.90 0.89 0.90 | 1.53 1.35 1.62 1.44 13.36 1.39 1.35 1.44 1.71 |
| Papa-ImageNet | StyleGAN2 [8] StyleGAN2 + DiffAug [8], [109] StyleGAN2 + ADA [110] StyleGAN2 + LeCam [8], [112] StyleGAN2 + APA [111] StyleGAN2 + D2D-CE [10] StyleGAN3-r + ADA [9] | 13.83 15.18 15.02 13.64 14.09 14.50 13.73 | 0.62 0.32 0.36 0.70 0.53 0.55 2.02 | 0.84 0.83 0.83 0.82 0.80 0.85 0.73 | 0.51 0.61 0.61 0.48 0.56 0.47 0.27 | 1.10 1.04 1.02 1.05 0.92 1.13 0.74 | 0.89 0.92 0.91 0.87 0.87 0.9 0.71 | 1.53 1.15 1.20 1.68 5.81 1.62 3.65 |
| Grandpa-ImageNet | SNGAN [107] SAGAN [49] | 17.85 14.52 | 5.79 6.42 | 0.87 0.87 | 0.03 0.02 | 1.48 1.58 | 0.45 0.40 | 10.87 12.02 |
| ImageNet | BigGAN [7] ContraGAN [85] ReACGAN [10] | 34.47 29.40 34.83 | 4.41 4.40 4.11 | 0.88 0.86 0.84 | 0.04 0.61 0.27 | 1.46 1.02 1.23 | 0.58 0.92 0.54 | 9.08 21.53 10.45 |
| AFHQv2 | StyleGAN2 [8] StyleGAN3-t [9] | 33.07 9.03 | 3.42 11.59 | 0.85 0.68 | 0.11 0.00 | 1.34 0.68 | 0.67 0.08 | 8.38 22.44 |
| FQ | SNGAN [107] SAGAN [49] | 12.65 10.26 | 6.36 6.90 | 0.89 0.88 | 0.02 0.02 | 1.76 1.80 | 0.46 0.41 | 11.59 12.65 |
| | BigGAN [7] ContraGAN [85] ReACGAN [10] | 19.81 19.72 22.47 | 5.24 4.54 4.22 | 0.91 0.89 0.89 | 0.02 0.03 0.03 | 1.99 1.81 1.84 | 0.58 0.59 0.62 | 10.16 19.72 11.00 |
| | StyleGAN2 [8] StyleGAN3-t [9] | 20.84 8.84 | 3.67 11.27 | 0.87 0.77 | 0.09 0.00 | 1.68 1.28 | 0.67 0.12 | 8.38 22.58 |
| | SNGAN [107] SAGAN [49] | 10.82 10.05 | 7.09 8.03 | 0.91 0.90 | 0.02 0.01 | 2.26 2.19 | 0.47 0.42 | 12.74 14.61 |
| | BigGAN [7] ContraGAN [85] ReACGAN [10] | 16.29 19.67 20.64 | 5.71 4.81 4.69 | 0.94 0.92 0.92 | 0.02 0.02 0.02 | 2.64 2.18 2.24 | 0.61 0.63 0.64 | 10.70 21.14 12.37 |
| | StyleGAN2 [8] StyleGAN3-t [9] | 17.38 7.30 | 4.08 13.28 | 0.88 0.80 | 0.08 0.00 | 1.87 1.25 | 0.67 0.09 | 9.20 25.74 |
| | SNGAN-256 [107] | 29.94 | 6.59 | 0.94 | 0.06 | 2.97 | 0.60 | 13.81 |
| | BigGAN-256 [7] ContraGAN-256 [85] ReACGAN-256 [10] | 46.14 31.97 89.28 | 4.55 5.94 4.32 | 0.94 0.93 0.94 | 0.15 0.07 0.05 | 3.16 2.92 2.66 | 0.75 0.62 0.73 | 13.10 17.25 16.51 |
| | StyleGAN2 [8] StyleGAN3-t [9] | 25.74 24.47 | 5.59 6.07 | 0.89 0.85 | 0.05 0.03 | 2.18 1.73 | 0.61 0.52 | 14.14 14.91 |
| | StyleGAN2 [8] StyleGAN2 + ADA [110] StyleGAN3-t + ADA [9] StyleGAN3-r + ADA [9] | 17.69 17.61 17.31 17.66 | 1.93 1.45 1.41 1.49 | 0.90 0.89 0.83 0.84 | 0.18 0.33 0.47 0.41 | 1.45 1.31 0.97 1.09 | 0.83 0.86 0.83 0.85 | 1.92 1.42 1.40 1.49 |
| | StyleGAN2 [8] | 3.75 | 0.54 | 0.86 | 0.36 | 1.31 | 0.90 | - |

TABLE A4. Benchmark table evaluated using **PyTorch-Swin-T** [37]. The resolutions of CIFAR10 [122], Baby/Papa/Grandpa ImageNet, ImageNet [106], AFHQv2 [9], [75], and FQ [54] datasets are 32, 64, 128, 512, and 1024, respectively. Top-1 and Top-2 performances are indicated in red and blue, respectively.

| PyTorch-Swin-T [37] | | TS ↑ | FTD ↓ | T-Precision ↑ | T-Recall ↑ | T-Density ↑ | T-Coverage ↑ | IFTD ↓ |
|---------------------|--------------------------------|--------------|--------------|---------------|-------------|-------------|--------------|---------------|
| CIFAR10 | DCGAN [1] | 3.16 | 75.08 | 0.33 | 0.10 | 0.36 | 0.17 | 143.96 |
| | LSGAN [147] | 3.40 | 68.57 | 0.30 | 0.16 | 0.27 | 0.20 | 140.83 |
| | Geometric GAN [148] | 2.98 | 66.14 | 0.40 | 0.09 | 0.44 | 0.19 | 135.38 |
| | ACGAN-Mod [79] | 3.27 | 57.61 | 0.42 | 0.07 | 0.46 | 0.24 | 103.57 |
| | WGAN [96] | 2.19 | 158.48 | 0.34 | 0.01 | 0.26 | 0.05 | 201.98 |
| | DRAGAN [98] | 3.23 | 58.42 | 0.38 | 0.23 | 0.42 | 0.21 | 129.11 |
| | WGAN-GP [52] | 2.87 | 91.45 | 0.32 | 0.13 | 0.30 | 0.15 | 155.69 |
| | PD-GAN [81] | 3.58 | 54.93 | 0.41 | 0.11 | 0.49 | 0.26 | 96.44 |
| | SNGAN [107] | 4.21 | 18.5 | 0.48 | 0.43 | 0.65 | 0.62 | 36.74 |
| | SAGAN [49] | 4.13 | 18.73 | 0.47 | 0.43 | 0.63 | 0.60 | 37.83 |
| | TACGAN [83] | 3.59 | 53.77 | 0.38 | 0.13 | 0.42 | 0.25 | 95.63 |
| | LOGAN [133] | 3.77 | 34.66 | 0.41 | 0.41 | 0.49 | 0.39 | 112.82 |
| | BigGAN [7] | 4.72 | 7.69 | 0.55 | 0.47 | 0.92 | 0.82 | 20.21 |
| | CRGAN [38] | 4.95 | 6.91 | 0.53 | 0.51 | 0.84 | 0.84 | 19.15 |
| | MHGAN [84] | 4.96 | 8.19 | 0.53 | 0.48 | 0.83 | 0.82 | 21.05 |
| Baby-ImageNet | ICRGAN [134] | 5.01 | 7.62 | 0.53 | 0.49 | 0.82 | 0.84 | 19.85 |
| | ContraGAN [85] | 4.66 | 10.39 | 0.54 | 0.46 | 0.85 | 0.76 | 136.36 |
| | BigGAN + DiffAug [109] | 4.48 | 6.82 | 0.59 | 0.46 | 1.03 | 0.86 | 18.05 |
| | BigGAN + LeCam [112] | 4.96 | 7.33 | 0.53 | 0.51 | 0.84 | 0.83 | 19.79 |
| | ADCGAN [87] | 4.76 | 7.83 | 0.53 | 0.50 | 0.84 | 0.82 | 20.80 |
| | ReACGAN [10] | 4.84 | 8.42 | 0.55 | 0.46 | 0.89 | 0.81 | 21.97 |
| | StyleGAN2 [8] | 5.04 | 10.65 | 0.51 | 0.48 | 0.82 | 0.80 | 24.31 |
| | StyleGAN2 + DiffAug [8], [109] | 5.16 | 6.07 | 0.55 | 0.50 | 0.96 | 0.88 | 17.55 |
| | StyleGAN2 + ADA [110] | 5.16 | 6.03 | 0.54 | 0.52 | 0.91 | 0.87 | 17.78 |
| | StyleGAN2 + LeCam [8], [112] | 4.86 | 12.01 | 0.50 | 0.45 | 0.84 | 0.79 | 26.38 |
| Papa-ImageNet | StyleGAN2 + APA [111] | 4.96 | 10.20 | 0.49 | 0.50 | 0.77 | 0.79 | 72.71 |
| | StyleGAN2 + D2D-CE [10] | 5.23 | 9.55 | 0.53 | 0.46 | 0.86 | 0.82 | 23.18 |
| | StyleGAN3-r + ADA [9] | 4.64 | 20.38 | 0.47 | 0.32 | 0.80 | 0.68 | 39.36 |
| | SNGAN [107] | 10.80 | 71.20 | 0.14 | 0.57 | 0.07 | 0.12 | 140.18 |
| | SAGAN [49] | 9.59 | 71.69 | 0.14 | 0.55 | 0.07 | 0.12 | 154.44 |
| Grandpa-ImageNet | BigGAN [7] | 13.72 | 36.66 | 0.32 | 0.39 | 0.29 | 0.35 | 90.85 |
| | ContraGAN [85] | 12.17 | 42.04 | 0.27 | 0.44 | 0.20 | 0.24 | 229.46 |
| | ReACGAN [10] | 13.71 | 34.98 | 0.31 | 0.40 | 0.24 | 0.29 | 100.35 |
| ImageNet | StyleGAN2 [8] | 12.76 | 50.31 | 0.25 | 0.55 | 0.20 | 0.32 | 111.18 |
| | StyleGAN3-t [9] | 6.32 | 172.08 | 0.01 | 0.06 | 0.00 | 0.00 | 322.42 |
| FQ | SNGAN [107] | 8.20 | 78.66 | 0.13 | 0.51 | 0.07 | 0.11 | 167.58 |
| | SAGAN [49] | 7.21 | 85.39 | 0.11 | 0.50 | 0.06 | 0.10 | 188.46 |
| AFHQv2 | BigGAN [7] | 9.54 | 52.72 | 0.23 | 0.38 | 0.19 | 0.25 | 128.80 |
| | ContraGAN [85] | 9.02 | 48.02 | 0.25 | 0.43 | 0.18 | 0.22 | 247.04 |
| | ReACGAN [10] | 9.98 | 43.34 | 0.25 | 0.42 | 0.19 | 0.25 | 133.91 |
| ImageNet | StyleGAN2 [8] | 10.49 | 58.18 | 0.17 | 0.55 | 0.12 | 0.22 | 135.01 |
| | StyleGAN3-t [9] | 5.78 | 150.59 | 0.05 | 0.02 | 0.02 | 0.01 | 323.48 |
| ImageNet | SNGAN [107] | 7.08 | 80.65 | 0.13 | 0.43 | 0.08 | 0.13 | 179.30 |
| | SAGAN [49] | 6.65 | 89.89 | 0.11 | 0.40 | 0.07 | 0.11 | 206.66 |
| | BigGAN [7] | 8.09 | 53.61 | 0.22 | 0.34 | 0.20 | 0.27 | 134.14 |
| ImageNet | ContraGAN [85] | 8.43 | 45.50 | 0.27 | 0.34 | 0.24 | 0.27 | 255.97 |
| | ReACGAN [10] | 9.34 | 43.35 | 0.26 | 0.32 | 0.23 | 0.29 | 147.42 |
| | StyleGAN2 [8] | 9.29 | 56.41 | 0.16 | 0.50 | 0.12 | 0.23 | 142.89 |
| ImageNet | StyleGAN3-t [9] | 4.30 | 161.33 | 0.10 | 0.00 | 0.05 | 0.02 | 345.20 |
| | SNGAN-256 [107] | 17.09 | 57.72 | 0.09 | 0.70 | 0.08 | 0.13 | 168.27 |
| ImageNet | BigGAN-256 [7] | 20.39 | 36.78 | 0.18 | 0.69 | 0.18 | 0.26 | 181.24 |
| | ContraGAN-256 [85] | 15.35 | 48.66 | 0.14 | 0.69 | 0.22 | 0.09 | 216.74 |
| | ReACGAN-256 [10] | 26.33 | 25.77 | 0.30 | 0.47 | 0.26 | 0.28 | 169.14 |
| ImageNet | StyleGAN2 [8] | 13.67 | 64.96 | 0.05 | 0.74 | 0.02 | 0.07 | 204.21 |
| | StyleGAN3-t [9] | 13.57 | 72.34 | 0.03 | 0.73 | 0.01 | 0.04 | 218.04 |
| AFHQv2 | StyleGAN2 [8] | 7.74 | 14.86 | 0.49 | 0.35 | 0.57 | 0.56 | 14.90 |
| | StyleGAN2 + ADA [110] | 8.23 | 12.78 | 0.53 | 0.40 | 0.71 | 0.63 | 12.99 |
| | StyleGAN3-t + ADA [9] | 7.47 | 14.57 | 0.40 | 0.54 | 0.42 | 0.55 | 14.59 |
| | StyleGAN3-r + ADA [9] | 7.75 | 16.61 | 0.35 | 0.56 | 0.35 | 0.53 | 16.56 |
| FQ | StyleGAN2 [8] | 4.02 | 16.15 | 0.57 | 0.44 | 0.85 | 0.77 | - |

TABLE A5. Benckmark table for GANs [7], [9], [10], [48], [63], autoregressive models [40], [41], [48], and diffusion probabilistic models [43], [44], [45], [46], [47]. We evaluate those generative models using **PyTorch-SwAV** [34] and the architecture-friendly resizer: PIL.BILINEAR for the postprocessing step. In the case of ImageNet-128 and ImageNet-256 experiments, we preprocess images using the best performing resizer among PIL.BILINEAR, PIL.BICUBIC, and PIL.LANCZOS based on Fréchet Distance. We mark \dagger if images are preprocessed using PIL.BILINEAR for evaluation, \ddagger for the case of PIL.BICUBIC, and $\ddagger\ddagger$ for the case of PIL.LANCZOS. Top-1 and Top-2 performances are indicated in red and blue, respectively.

| | Model | SS \uparrow | FSD \downarrow | S-Precision \uparrow | S-Recall \uparrow | S-Density \uparrow | S-Coverage \uparrow | Avg. Rank |
|--------------|--|---------------|------------------|------------------------|---------------------|----------------------|-----------------------|-------------|
| CIFAR10 | ReACGAN + DiffAug (Ours) [10] | 14.28 | 0.37 | 0.85 | 0.55 | 1.09 | 0.90 | 8.33 |
| | StyleGAN2-ADA [85] | 14.57 | 0.36 | 0.84 | 0.59 | 1.08 | 0.92 | 7.50 |
| | StyleGAN2-ADA (Ours) [85] | 15.30 | 0.33 | 0.83 | 0.63 | 1.05 | 0.92 | 6.50 |
| | StyleGAN2 + DiffAug + D2D-CE (Ours) [10] | 14.79 | 0.34 | 0.87 | 0.58 | 1.19 | 0.94 | 5.50 |
| | DDPM [43] | 13.47 | 0.49 | 0.89 | 0.62 | 1.35 | 0.93 | 6.17 |
| | DDPM++ [44] | 14.03 | 0.33 | 0.88 | 0.67 | 1.27 | 0.95 | 4.00 |
| | NCSN++ [44] | 15.03 | 0.33 | 0.83 | 0.73 | 0.97 | 0.92 | 6.17 |
| | LSGM [45] | 14.78 | 0.45 | 0.88 | 0.69 | 1.20 | 0.94 | 4.50 |
| | LSGM-ODE [45] | 14.67 | 0.27 | 0.85 | 0.74 | 1.06 | 0.94 | 4.33 |
| | CLD-SGM [47] | 14.27 | 0.30 | 0.88 | 0.69 | 1.25 | 0.95 | 3.50 |
| ImageNet-128 | StyleGAN-XL [63] | 16.00 | 0.24 | 0.84 | 0.45 | 1.06 | 0.91 | 6.50 |
| | BigGAN [7] $\dagger\ddagger$ | 128.55 | 4.56 | 0.98 | 0.07 | 3.76 | 0.83 | 3.83 |
| | BigGAN (Ours) [7] \dagger | 97.27 | 4.49 | 0.97 | 0.12 | 3.36 | 0.81 | 4.17 |
| | BigGAN-Deep [7] $\dagger\ddagger$ | 230.57 | 2.49 | 0.97 | 0.17 | 2.89 | 0.91 | 2.83 |
| | ReACGAN (Ours) [10] \dagger | 131.14 | 3.18 | 0.94 | 0.10 | 2.40 | 0.83 | 4.17 |
| | StyleGAN-XL [63] $\dagger\ddagger\ddagger$ | 311.54 | 1.04 | 0.91 | 0.39 | 1.68 | 0.94 | 2.83 |
| ImageNet-256 | ADM-G [46] \dagger | 233.15 | 1.31 | 0.92 | 0.51 | 1.66 | 0.97 | 2.83 |
| | BigGAN [7] \dagger | 202.85 | 4.02 | 0.98 | 0.10 | 3.35 | 0.85 | 5.50 |
| | BigGAN-Deep [7] \dagger | 271.73 | 3.57 | 0.98 | 0.07 | 3.43 | 0.87 | 4.83 |
| | VQGAN [48] \dagger | 323.16 | 3.03 | 0.91 | 0.25 | 1.62 | 0.84 | 6.00 |
| | StyleGAN-XL [63] $\dagger\ddagger$ | 383.64 | 1.08 | 0.90 | 0.40 | 1.60 | 0.94 | 4.17 |
| | ADM-G [46] \dagger | 320.47 | 1.54 | 0.92 | 0.48 | 1.59 | 0.95 | 3.83 |
| | ADM-G-U [46] \dagger | 322.29 | 1.78 | 0.93 | 0.42 | 1.81 | 0.95 | 3.33 |
| | MaskGIT [40] \dagger | 265.34 | 2.48 | 0.96 | 0.27 | 2.42 | 0.95 | 4.00 |
| ImageNet-256 | RQ-Transformer [41] \dagger | 390.48 | 2.14 | 0.95 | 0.36 | 1.87 | 0.93 | 3.67 |

TABLE A6. Benckmark table for GANs [7], [9], [10], [48], [63], autoregressive models [40], [41], [48], and diffusion probabilistic models [43], [44], [45], [46], [47]. We evaluate those generative models using **PyTorch-Swin-T** [37] and the architecture-friendly resizer: PIL.BICUBIC for the postprocessing step. In the case of ImageNet-128 and ImageNet-256 experiments, we preprocess images using the best performing resizer among PIL.BILINEAR, PIL.BICUBIC, and PIL.LANCZOS based on Fréchet Distance. We mark \dagger if images are preprocessed using PIL.BILINEAR for evaluation, \ddagger for the case of PIL.BICUBIC, and $\ddagger\ddagger$ for the case of PIL.LANCZOS. Top-1 and Top-2 performances are indicated in red and blue, respectively.

| | Model | TS \uparrow | FTD \downarrow | T-Precision \uparrow | T-Recall \uparrow | T-Density \uparrow | T-Coverage \uparrow | Avg. Rank |
|--------------|--|---------------|------------------|------------------------|---------------------|----------------------|-----------------------|-------------|
| CIFAR10 | ReACGAN + DiffAug (Ours) [10] | 4.91 | 6.14 | 0.55 | 0.48 | 0.88 | 0.83 | 9.33 |
| | StyleGAN2-ADA [85] | 5.11 | 6.29 | 0.55 | 0.51 | 0.92 | 0.87 | 7.67 |
| | StyleGAN2-ADA (Ours) [85] | 5.12 | 5.43 | 0.57 | 0.50 | 1.00 | 0.89 | 5.83 |
| | StyleGAN2 + DiffAug + D2D-CE (Ours) [10] | 5.14 | 6.64 | 0.57 | 0.49 | 0.94 | 0.88 | 7.00 |
| | DDPM [43] | 5.14 | 7.30 | 0.55 | 0.55 | 0.87 | 0.85 | 7.33 |
| | DDPM++ [44] | 5.01 | 4.56 | 0.59 | 0.55 | 1.02 | 0.90 | 4.50 |
| | NCSN++ [44] | 4.84 | 3.54 | 0.62 | 0.54 | 1.11 | 0.92 | 4.17 |
| | LSGM [45] | 4.39 | 5.93 | 0.68 | 0.48 | 1.34 | 0.93 | 5.00 |
| | LSGM-ODE [45] | 4.97 | 3.40 | 0.59 | 0.57 | 0.97 | 0.92 | 4.00 |
| | CLD-SGM [47] | 5.00 | 3.75 | 0.59 | 0.55 | 0.99 | 0.91 | 4.67 |
| ImageNet-128 | StyleGAN-XL [63] | 5.05 | 2.99 | 0.62 | 0.43 | 1.13 | 0.92 | 3.83 |
| | BigGAN [7] \dagger | 37.04 | 17.91 | 0.34 | 0.48 | 0.42 | 0.58 | 4.33 |
| | BigGAN (Ours) [7] \dagger | 26.90 | 16.72 | 0.32 | 0.56 | 0.38 | 0.57 | 4.67 |
| | BigGAN-Deep [7] $\dagger\ddagger$ | 53.71 | 9.82 | 0.49 | 0.46 | 0.76 | 0.73 | 3.17 |
| | ReACGAN (Ours) [10] \dagger | 29.55 | 15.05 | 0.34 | 0.47 | 0.37 | 0.46 | 5.00 |
| | StyleGAN-XL [63] $\dagger\ddagger\ddagger$ | 60.72 | 4.47 | 0.50 | 0.50 | 0.72 | 0.84 | 2.00 |
| ImageNet-256 | ADM-G [46] \dagger | 52.88 | 6.03 | 0.51 | 0.55 | 0.82 | 0.89 | 1.67 |
| | BigGAN [7] \dagger | 111.25 | 19.15 | 0.39 | 0.49 | 0.50 | 0.54 | 7.17 |
| | BigGAN-Deep [7] \dagger | 125.19 | 13.93 | 0.43 | 0.43 | 0.55 | 0.58 | 6.17 |
| | VQGAN [48] $\dagger\ddagger$ | 198.75 | 11.66 | 0.40 | 0.53 | 0.40 | 0.57 | 5.17 |
| | StyleGAN-XL [63] $\dagger\ddagger$ | 159.50 | 5.43 | 0.48 | 0.52 | 0.62 | 0.77 | 3.00 |
| | ADM-G [46] $\dagger\ddagger$ | 131.31 | 8.37 | 0.49 | 0.59 | 0.61 | 0.77 | 3.00 |
| | ADM-G-U [46] $\dagger\ddagger$ | 174.74 | 7.81 | 0.54 | 0.55 | 0.71 | 0.81 | 1.83 |
| | MaskGIT [40] \dagger | 108.39 | 13.33 | 0.35 | 0.63 | 0.36 | 0.62 | 6.00 |
| | RQ-Transformer [41] \dagger | 215.46 | 9.20 | 0.52 | 0.52 | 0.59 | 0.71 | 3.33 |

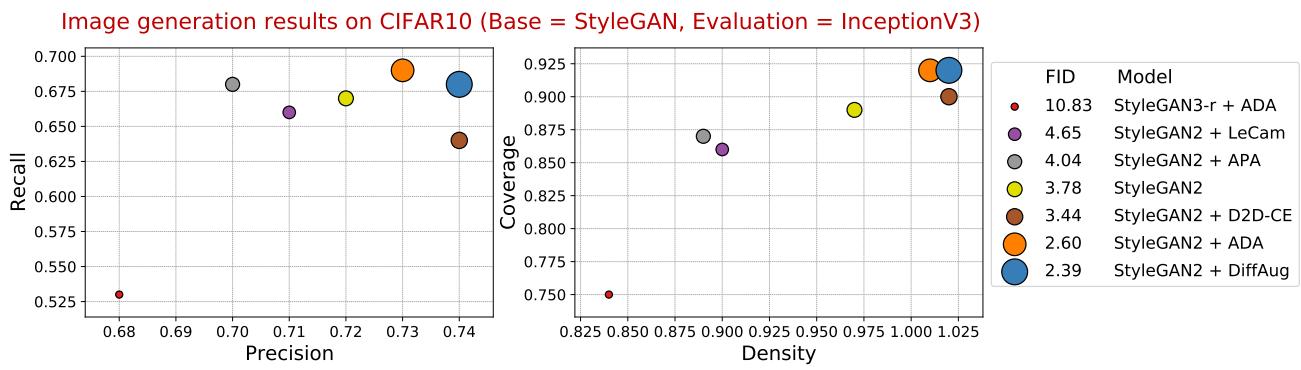
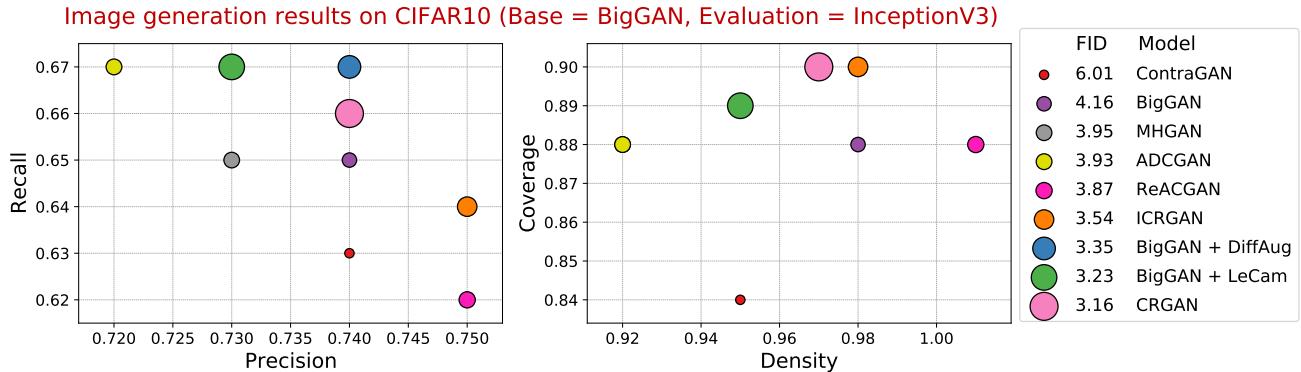
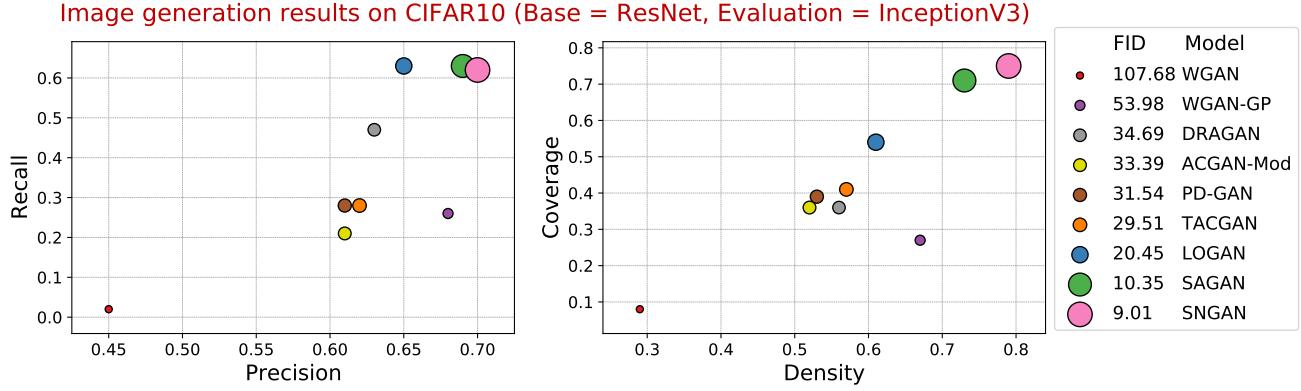


Figure A3. Scatter plots for analyzing GANs from the perspective of fidelity and diversity. We adopt TensorFlow-InceptionV3 [30] as the evaluation backbone.

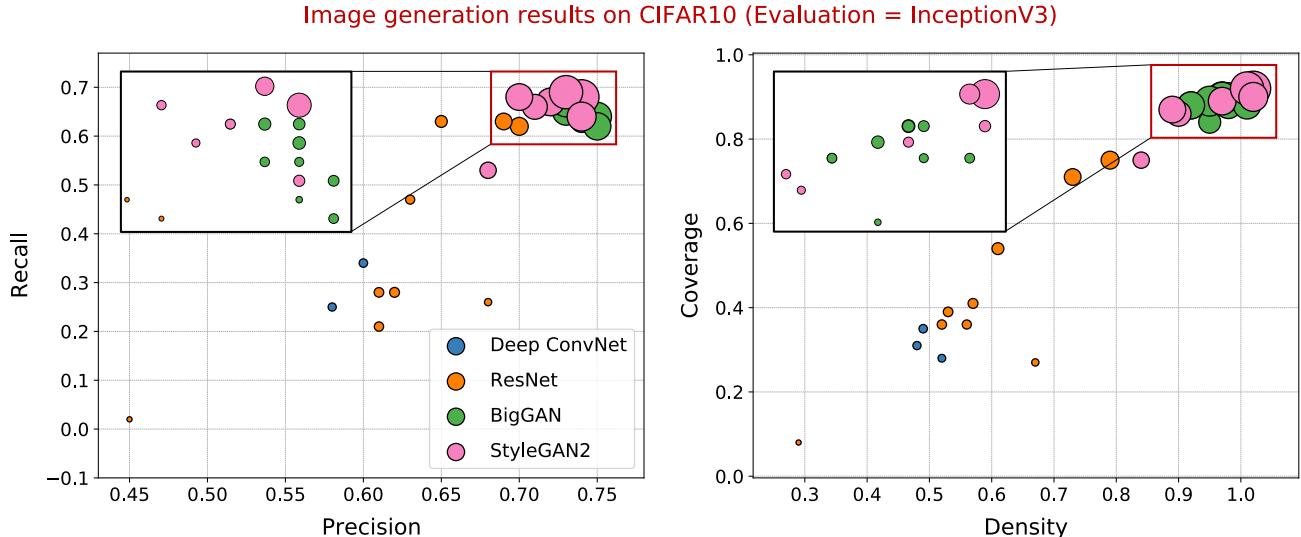


Figure A4. Scatter plots for analyzing the characteristics of distinct GAN architectures. We train GANs on CIFAR10 [122] and evaluate each GAN using TensorFlow-InceptionV3 [30].

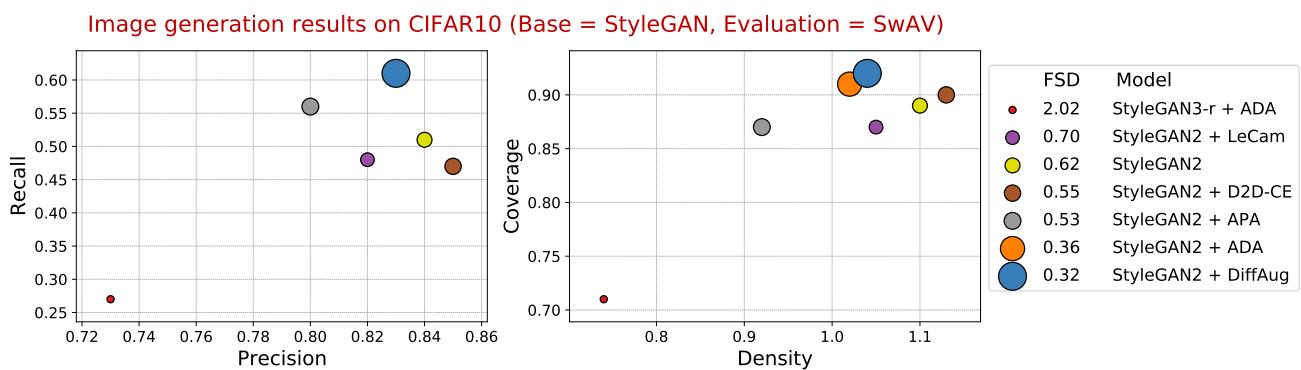
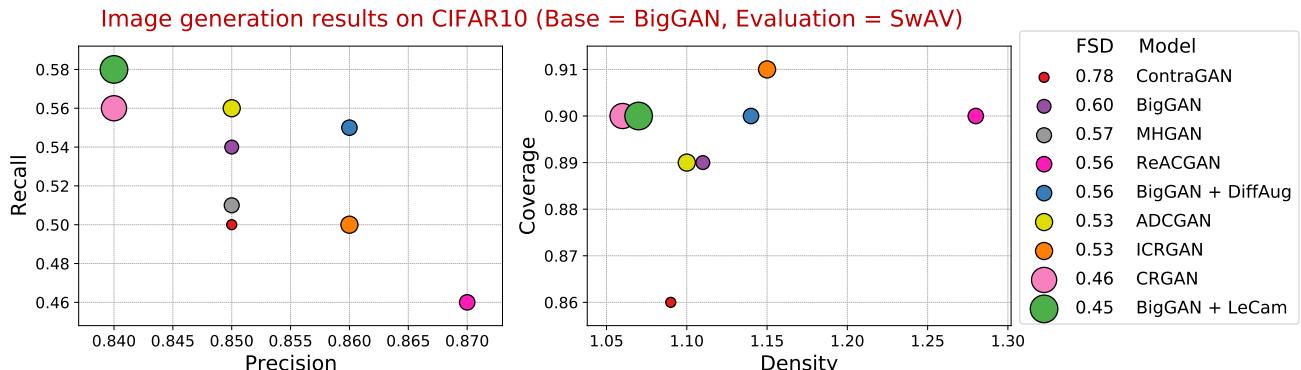
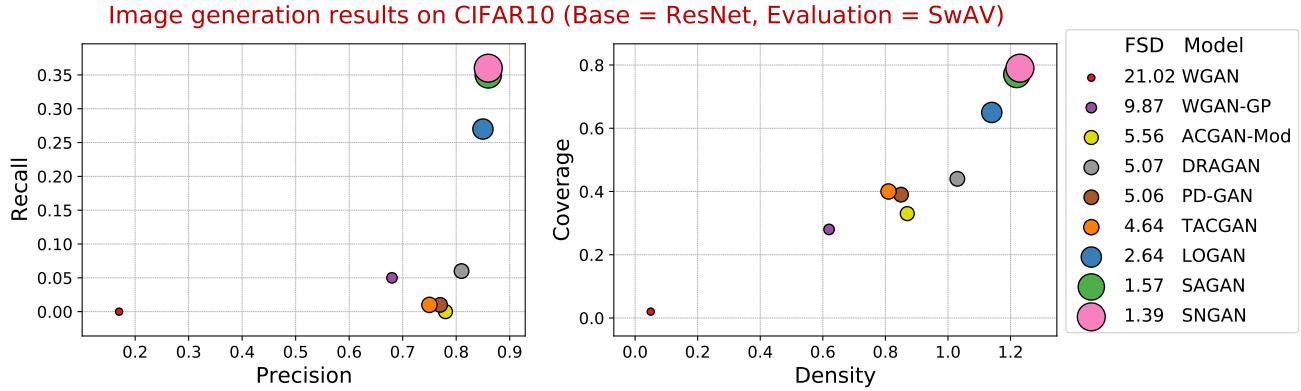


Figure A5. Scatter plots for analyzing GANs from the perspective of fidelity and diversity. We adopt PyTorch-SwAV [34] as the evaluation backbone.

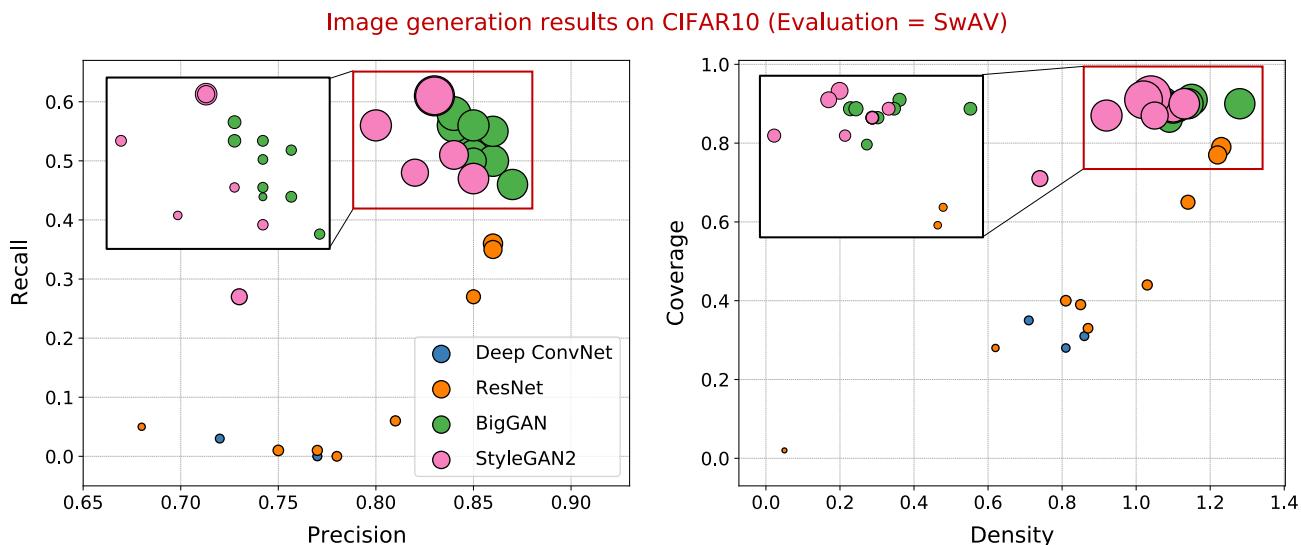


Figure A6. Scatter plots for analyzing the characteristics of distinct GAN architectures. We train GANs on CIFAR10 [122] dataset and evaluate each GAN using SwAV [34] network.

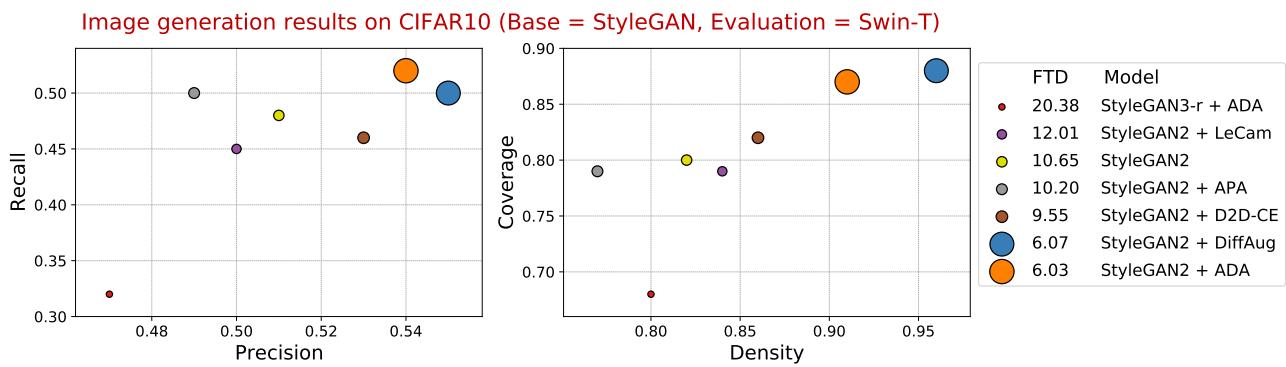
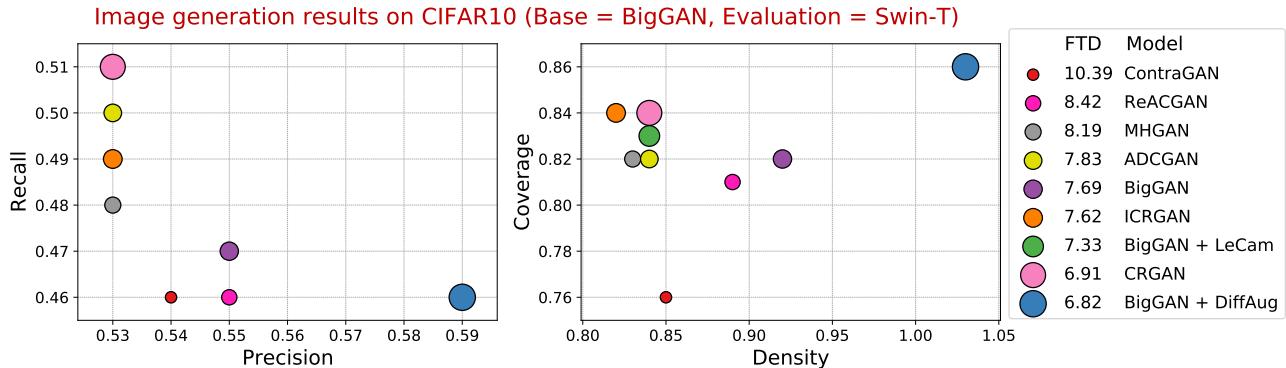
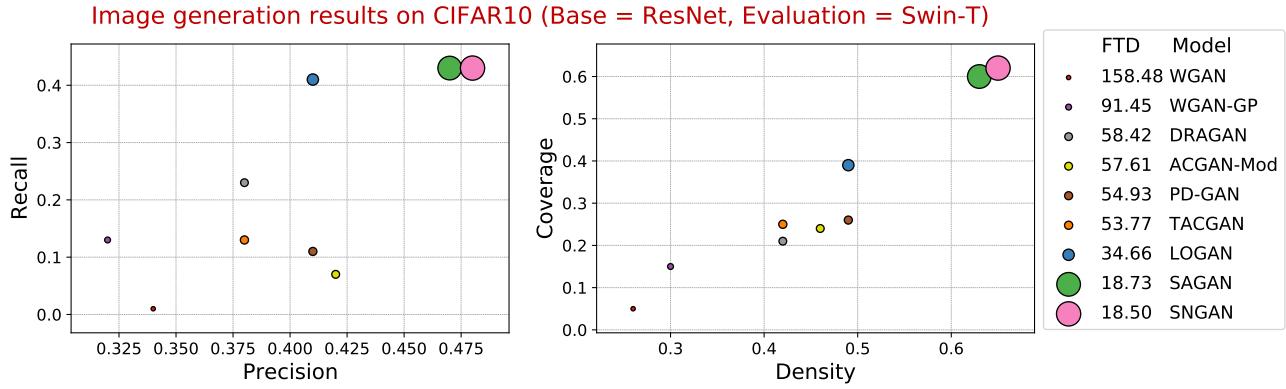


Figure A7. Scatter plots for analyzing GANs from the perspective of fidelity and diversity. We use PyTorch-Swin-T [37] as the evaluation backbone.

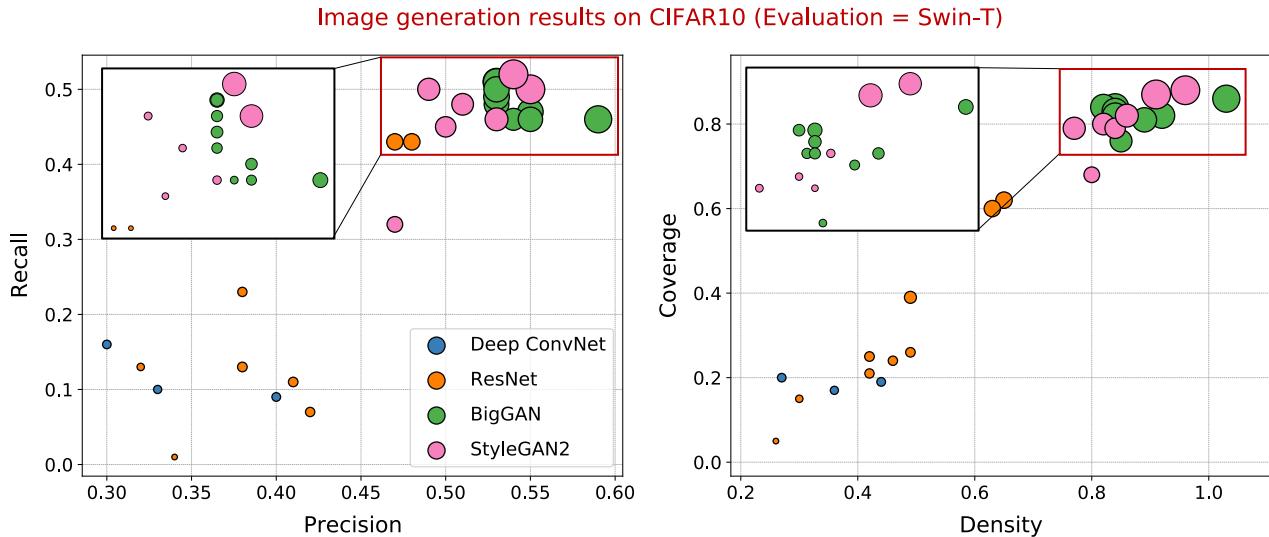


Figure A8. Scatter plots for analyzing the characteristics of distinct GAN architectures. We train GANs on CIFAR10 [122] dataset and evaluate each GAN using Swin-T [37] network.

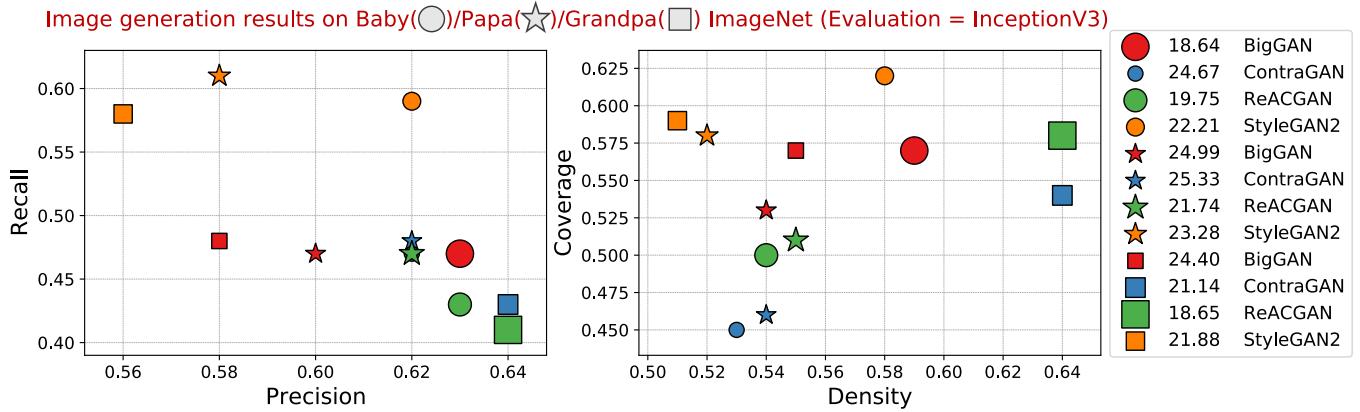


Figure A9. Scatter plots for analyzing the characteristics of distinct GAN architectures. We train GANs on Baby, Papa, and Grandpa ImageNet datasets and evaluate those GANs using TensorFlow-InceptionV3 [30].

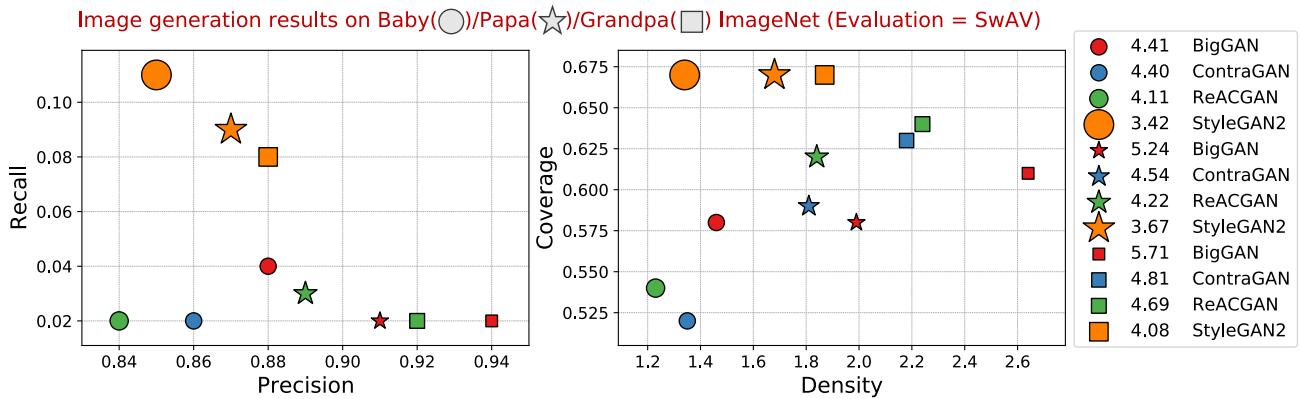


Figure A10. Scatter plots for analyzing the characteristics of distinct GAN architectures. We train GANs on Baby, Papa, and Grandpa ImageNet datasets and evaluate those GANs using PyTorch-SwAV [34].

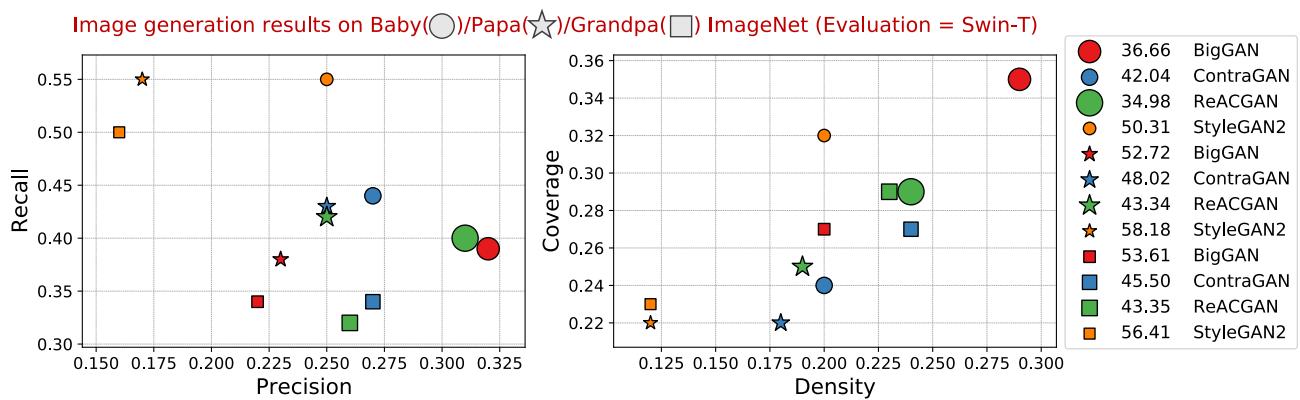


Figure A11. Scatter plots for analyzing the characteristics of distinct GAN architectures. We train GANs on Baby, Papa, and Grandpa ImageNet datasets and evaluate those GANs using PyTorch-Swin-T [37].



Figure A12. Random selection of generated images on ImageNet-256 [106] using StyleGAN-XL [63] (FID = 2.32, FSD = 1.08, and FTD = 5.43).



Figure A13. Random selection of generated images on ImageNet-256 [106] using RQ-Transformer [41] (FID = 3.83, FSD = 2.14, and FTD = 9.2).



Figure A14. Random selection of generated images on ImageNet-256 [106] using ADM-G-U [46] (FID = 4.01, FSD = 1.78, and FTD = 7.81).