A Novel Generator With Auxiliary Branch for Improving GAN Performance

Seung Park and Yong-Goo Shin, *Member, IEEE*

***Abstract*— The generator in the generative adversarial network (GAN) learns image generation in a coarse-to-fine manner in which earlier layers learn the overall structure of the image and the latter ones refine the details. To propagate the coarse information well, recent works usually build their generators by stacking up multiple residual blocks. Although the residual block can produce a high-quality image as well as be trained stably, it often impedes the information flow in the network. To alleviate this problem, this brief introduces a novel generator architecture that produces the image by combining features obtained through two different branches: the main and auxiliary branches. The goal of the main branch is to produce the image by passing through the multiple residual blocks, whereas the auxiliary branch is to convey the coarse information in the earlier layer to the later one. To combine the features in the main and auxiliary branches successfully, we also propose a gated feature fusion module (GFFM) that controls the information flow in those branches. To prove the superiority of the proposed method, this brief provides extensive experiments using various standard datasets including CIFAR-10, CIFAR-100, LSUN, CelebA-HQ, AFHQ, and tiny-ImageNet. Furthermore, we conducted various ablation studies to demonstrate the generalization ability of the proposed method. Quantitative evaluations prove that the proposed method exhibits impressive GAN performance in terms of Inception score (IS) and Frechet inception distance (FID). For instance, the proposed method boosts the FID and IS scores on the tiny-ImageNet dataset from 35.13 to 25.00 and 20.23 to 25.57, respectively.**

***Index Terms*— Adversarial learning, auxiliary branch, gated feature fusion module (GFFM), generative adversarial network (GAN), image generation.**

I. Introduction

Generative adversarial network (GAN) [1], is an algorithmic framework that synthesizes a target data distribution from a given prior distribution. GAN has attracted great attention by achieving great success for various applications including text-to-image translation [2], [3], image-to-image translation [4], [5], [6], and image inpainting [7], [8]. In general, GAN is composed of two different networks called generator and discriminator. The goal of the generator is to produce the real data distribution to fool the discriminator, whereas the discriminator is trained to classify the real sample from the fake one which is synthesized by the generator. Since the goals of the generator and discriminator are opposed to each other, GAN is difficult to train stably compared with the supervised learning-based convolutional neural networks (CNNs).

Manuscript received 30 December 2021; revised 27 May 2023 and 30 September 2023; accepted 28 January 2024. Date of publication 12 February 2024; date of current version 1 March 2025. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government Ministry of Science and Information and Communication Technology (MSIT) under Grant 2022R1G1A1004001; in part by MSIT, South Korea under the Information Technology Research Center (ITRC) support Program under Grant IITP-2023-RS-2023-00258971; and in part by the Institute for Information and Communications Technology Planning and Evaluation (IITP). *(Corresponding author: Yong-Goo Shin.)*

Seung Park is with the Department of Biomedical Engineering, Chungbuk National University Hospital, Chungbuk National University College of Medicine, Cheongju-si, Chungcheongbuk-do 28644, Republic of Korea (e-mail: spark.cbnuh@gmail.com).

Yong-Goo Shin is with the Department of Electronics and Information Engineering, Korea University, Sejong-si 30019, Republic of Korea (e-mail: ygshin92@korea.ac.kr).

Digital Object Identifier 10.1109/TNNLS.2024.3361087

Recent studies observed that the sharp gradient space of the discriminator is the major reason for the unstable problem in the GAN training [9]. To resolve this problem, some briefs [10], [11], [12] introduced regularization or normalization techniques that inhibit the discriminator from making the sharp gradient space. As the regularization method, the conventional methods [11], [13], [14] usually added the regularization term into the adversarial loss function. Among the various regularization approaches, gradient penalty-based regularization methods [11], [13], [14], which add the gradient norm as the penalty term, are the most popular techniques and have been widely used. The normalization methods [10], [15], [16] also have been widely studied. Spectral normalization (SN) [10] is the most popular one that controls the Lipschitz constraint on the discriminator by constraining the spectral norm of each layer. Since these regularization and normalization methods can lead the stable GAN training, recent studies usually employ these techniques when training their networks.

Some researchers [17], [18], [19], [20] studied architectural modules to improve the GAN performance. Miyato and Koyama [18] introduced a conditional projection technique that provides conditional information to the discriminator by projecting the conditional weight vector to the feature vector. This method significantly boosts the performance of the conditional GAN (cGAN) scheme. Zhang et al. [19] introduced a self-attention module that guides the generator and discriminator on where to attend. Yeo et al. [20] introduced a simple yet effective technique, called a cascading rejection module, which produces dynamic features in an iterative manner at the last layer of the discriminator. Li et al. [21] proposed a dynamic weighted GAN (DW-GAN) having multiple discriminators, which synthesizes the images following the color tone of the target image. These methods generate images with plausible quality but still use the common generator architecture.

On the other hand, some briefs investigated new forms of convolution layer to improve the GAN performance. Park et al. [22] proposed a perturbed convolution that prevents the discriminator from falling into the overfitting problem. However, since PConv is developed for the discriminator, it is hard to apply to the generator. Sagong et al. [23] introduced a conditional convolution layer for the generator, which incorporates the conditional information into the convolution operation. Although this method shows fine performance on cGAN, it has a limitation: this method only can be used for cGAN scheme since it is designed to replace the conditional batch normalization (BN). Recently, Park and Shin [24] proposed a novel convolution layer, called a generative convolution, which controls the convolution weight of the generator according to the given latent vector.

There have been several attempts to newly design the generator and discriminator architectures for boosting the GAN performance. Specifically, in the early stage of the GAN study, Radford et al. [25] designed the generator and discriminator architectures by stacking the transposed convolution layer and standard convolution layer in series, respectively. These network architectures are simple to build but generate low-quality images. Karras et al. [26] introduced the GAN with a novel training strategy, called a progressive growing skill, which effectively leads the generator and discriminator to produce

high-resolution images. Recently, Miyato et al. [10], [18] proposed a spectrally normalized GAN (SNGAN) that builds generator and discriminator by stacking multiple residual blocks [27]. To further improve the SNGAN performance, Brock et al. [28] proposed a new generative model, called BigGAN, which has a SNGAN-like structure in which noise is additionally input to the residual block. Since SNGAN and BigGAN are easy to implement and train stably, most recent studies employ these networks as their baseline networks. However, even though SNGAN and BigGAN exhibit fine GAN performance, we believe that designing a novel generator and discriminator architectures for improving the GAN performance still remains an open question. Therefore, this brief mainly focuses on developing the novel generator architecture.

A common characteristic of generators in SNGAN and BigGAN is that they are built by stacking multiple residual blocks in series. These generators produce images in a coarse-to-fine manner, where the earlier blocks provide a rough template and the latter ones clarify the details. However, as pointed out in [29], even if the residual block is used, the information flow in the generator may be impeded. That means it may be difficult to impart coarse information in the earlier blocks to the latter ones. To alleviate this issue, this brief proposes a novel generator architecture that embraces the strength point of the conventional methods as well as further improves the GAN performance. The proposed method consists of two different branches: the main and auxiliary branches. The main branch learns features by passing through the multiple residual blocks, whereas the auxiliary branch propagates the coarse information in the earlier residual block to the later one. In other words, the goal of the auxiliary branch is to assist the information flow in the generator. In order to combine the features in both branches effectively, we introduce a gated feature fusion module (GFFM) that controls the information flow in those branches. To demonstrate the superiority of the proposed method, this brief presents extensive experimental results using various standard datasets such as CIFAR-10 [30], CIFAR-100 [30], LSUN [31], CelebA-HQ [26], AFHQ [32], and tiny-ImageNet [33], [34]. Furthermore, we provided various ablation studies to demonstrate the generalization ability of the proposed method. Quantitative evaluations show that the proposed method significantly improves GAN performance in terms of Inception score (IS) and Frechet inception distance (FID).

Key contributions of our brief are as follows. First, we propose a novel generator architecture that contains the auxiliary branch and GFFM, which support the information flow in the generator. Second, with slight additional hardware costs, the proposed method significantly improves the GAN and cGAN performances on various datasets in terms of FID and IS scores. For instance, the proposed method improves FID and IS scores on the tiny-ImageNet dataset from 35.13 to 25.00 and 20.23 to 25.57, respectively.

II. Background

A. Generative Adversarial Network

GAN [1] is a min-max game between two different networks called generator G and discriminator D. G is trained to generate visually pleasing images, whereas D is optimized to classify the generated images from real ones. However, both networks compete with each other, GAN is hard to train stably compared with supervised learning. To address this problem, some studies [11], [15], [35], [36] have been conducted to reformulate the objective function. For instance, Mao et al. [35] introduced the objective function using the least square errors, whereas Arjovsky et al. [15] proposed Wasserstein GAN (WGAN) that computes the Wasserstein distance between the real and generated images. Another notable objective function is a hinge-adversarial loss [36], which is widely used in practice [7], [8],

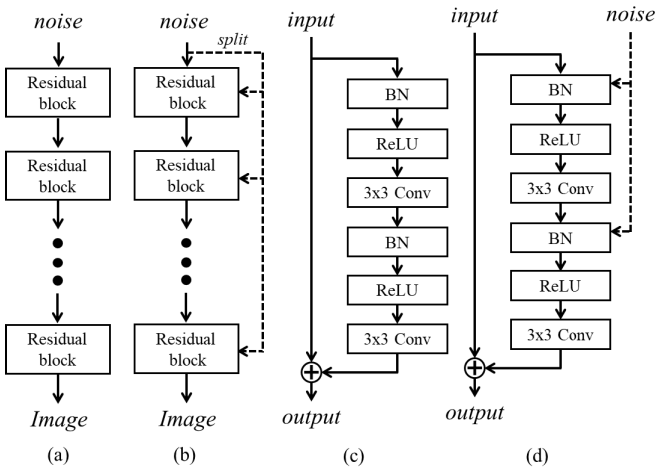

Fig. 1. Overall generator architectures and residual blocks utilized in SNGAN [10], [18] and BigGAN [28]. (a) Generator architecture for SNGAN. (b) Generator architecture for BigGAN. (c) Residual block for SNGAN. (d) Residual block for BigGAN.

[10], [17], [18], [20], [22], [24], [28]. On the other hand, cGAN, which produces class-conditional images, has also been actively studied [18], [37], [38]. In general, cGAN utilizes additional conditional information such as class labels or text conditions, in order to produce the condition-specific feature. By optimizing the objective functions for cGAN [18], [37], the generator can select the image class to be generated.

B. Residual Block-Based Generator

In the field of GAN-based image generation, SNGAN [10], [18] and BigGAN [28] are the most popular base networks since they are simple and exhibit fine performance. The overall generator architectures of those methods are depicted in Fig. 1. More specifically, in the SNGAN paper, they built the generator using the multiple standard residual blocks. To further improve the GAN performance, Brock et al. [28] introduced a residual block in which the noise vector is additionally input to infer the scaling and shifting parameters of the BN layer. This approach slightly modifies the residual block but still has a similar generator architecture to SNGAN. Indeed, as observed in the previous studies [39], [40], the generator produces images in a coarse-to-fine manner, where the earlier layers provide a rough template and the latter layers refine the details. That means it needs to convey the information in the earlier layers to the latter ones well. However, since the generators in SNGAN and BigGAN only contain the multiple residual blocks, it may impede the information flow in the network [29].

To validate our hypothesis, we conducted simple experiments: we replaced the residual block with the dense block [29] which propagates the previous information to the latter much better. In our experiments, we used BigGAN as the baseline model and evaluated the GAN performance in terms of IS and FID on the CIFAR-10 [30] and CIFAR-100 [30] datasets. The detailed descriptions of the network architectures and evaluation metrics will be explained in Sections III-B and IV-A, respectively. As shown in Table I summarizing the experimental results, when replacing the residual block with the dense block, the GAN performances are improved on both datasets. In addition, to match the number of network parameters, we conducted ablation studies that increase the channel dimensions of the residual block (Residual* in Table I). Although Residual* shows better performance than the residual block, it is

TABLE I
COMPARISON OF THE DENSE BLOCK WITH RESIDUAL ONE ON THE CIFAR-10 AND CIFAR-100 DATASETS IN TERMS OF IS AND FID. THE BEST RESULTS ARE IN BOLD

Block name		Residual		Residual*		Dense	
Parameters		3.78M		4.74M		4.74M	
Dataset		IS↑	FID↓	IS↑	FID↓	IS↑	FID↓
CIFAR-10	trial 1	7.70	13.98	7.74	12.93	7.96	12.41
	trial 2	7.80	13.69	7.68	13.34	7.95	12.20
	trial 3	7.87	13.09	7.79	12.93	7.92	12.04
	Mean	7.79	13.58	7.74	13.07	**7.94**	**12.22**
	Std	0.08	0.45	0.05	0.24	**0.02**	**0.18**
CIFAR-100	trial 1	7.99	17.45	7.99	17.22	8.22	15.35
	trial 2	7.94	17.72	7.97	16.78	8.07	16.74
	trial 3	7.94	17.57	8.02	16.86	8.15	15.73
	Mean	7.96	17.58	8.00	16.95	**8.15**	**15.94**
	Std	0.03	0.13	0.03	0.24	**0.08**	**0.72**

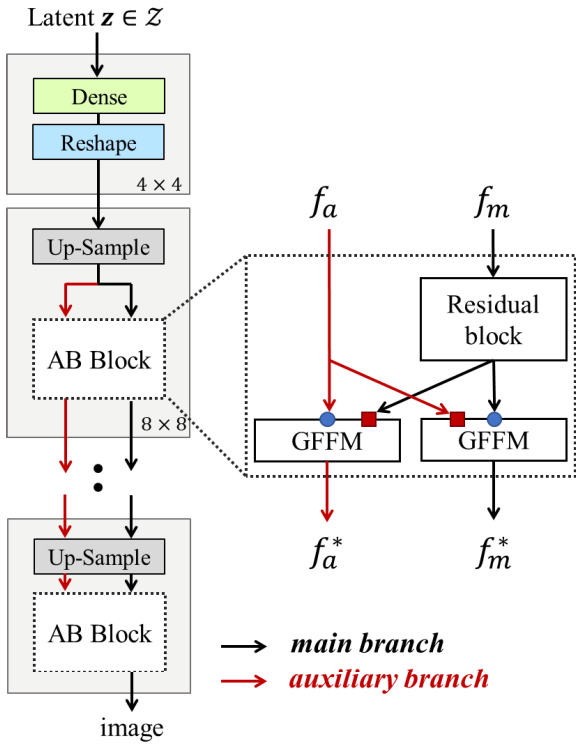

Fig. 2. Overall generator architecture of the proposed method. The proposed method builds the generator using multiple AB blocks which contain the main and auxiliary branches.

still weak compared with the dense block. That means even with the same number of network parameters, the dense block is more effective in achieving fine GAN performance than the residual block. Based on these results, we expect that the GAN performance could be improved by propagating the information in the earlier layers to the latter ones well.

III. PROPOSED METHOD

A. Generator With Auxiliary Branch

This brief introduces the novel generator architecture including main and auxiliary branches. Fig. 2 illustrates the layout of the proposed method schematically. The goal of the auxiliary branch is to convey the coarse information in the earlier layers to the latter ones. Indeed, an intuitive way to flow information well is to directly sum or concatenate features in the earlier layers to the later ones. However, we should note that the feature levels in earlier and later layers are different. More specifically, the earlier layers produce high-dimensional features that contain coarse information,

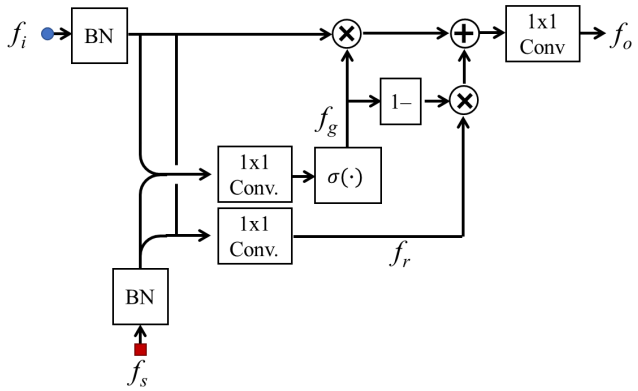

Fig. 3. Detailed architecture of the proposed GFFM. BN and Conv are the BN [42] and convolution operation, respectively. In addition, $\sigma(\cdot)$ indicates the sigmoid function.

whereas the latter ones learn low-dimensional features for image details. Therefore, it may not be the best choice to directly sum or concatenate those features. In addition, the latter layers do not need all information in the earlier layers; they may only need some parts of the information in the earlier layers. That means some procedures are required to select the appropriate features and combine them successfully.

To this end, inspired by the gated recurrent unit (GRU) [41], we design the GFFM that integrates features in both branches. In GRU, the two different features, i.e., hidden and the current features, are combined by using the gating mechanism, which leads the fine performance in the field of natural language processing. Following this approach, we build the GFFM architecture that combines the features from the main and auxiliary branches using the gating mechanism. The architecture of GFFM is depicted in Fig. 3, where f_m and f_a indicate the features in the main and auxiliary branches, respectively. In the proposed method, since f_m and f_a have different levels of the features, we first utilize BN [42] to balance the scales of the features. Then, we blend the normalized features using the gating unit that controls the information flow through a gating mechanism. Specifically, the gating unit needs an input feature $f_i \in \mathbb{R}^{h\times w\times c_i}$ which will be refined, and requires a side feature $f_s \in \mathbb{R}^{h\times w\times c_s}$ that contains the information to be used for f_i refinement. Here, h and w indicate the height and width of feature maps, respectively, whereas c_x means the channel numbers of f_x. By using f_i and f_s, the gate feature $f_g \in \mathbb{R}^{h\times w\times c_g}$, which controls the information passed on in the hierarchy is computed as follows:

$$f_g = \sigma\left[W_g * (f_i \odot f_s)\right] \tag{1}$$

where $*$ and $\odot$ indicate the convolution and channel-concatenation operations, respectively, whereas W_g is a trainable weight in the convolution layer. σ is a sigmoid function that produces output in the range [0, 1] to control how much to forget or keep the information. In addition, we generate the refinement feature $f_r \in \mathbb{R}^{h\times w\times c_r}$, which will be newly added to f_i, in a similar way. The f_r is defined as

$$f_r = \left[W_f * (f_i \odot f_s)\right] \tag{2}$$

where W_f is a trainable weight in the convolution layer. By using these features, i.e., f_g and f_r, we produce the output feature $f_o \in \mathbb{R}^{h\times w\times c_o}$ as

$$f_o = W_o * \left[f_g \otimes f_i + \left(1 - f_g\right) \otimes f_r\right] \tag{3}$$

where $\otimes$ indicates element-wise multiplication and W_o is a weight in the output convolution layer. Here, we could interpret the (3) as follows: the first term $f_g \otimes f_i$ means that the gating unit decides what

TABLE II

NETWORK ARCHITECTURES OF THE DISCRIMINATOR (TOP) AND GENERATOR (BOTTOM) FOR EACH IMAGE RESOLUTION. THE RB MEANS THE RESIDUAL BLOCK PRESENTED IN FIG. 1(D)

32×32 resolution	128×128 resolution	256×256 resolution	512×512 resolution
RGB image	RGB image	RGB image	RGB image
RB, down, 128	RB, down, 64	RB, down, 32	RB, down, 16
RB, down, 128	RB, down, 128	RB, down, 64	RB, down, 32
RB, 128	RB, down, 256	RB, down, 128	RB, down, 64
RB, 128	RB, down, 512	RB, down, 256	RB, down, 128
ReLU	RB, down, 512	RB, down, 512	RB, down, 256
Global sum pooling	RB, 512	RB, down, 512	RB, down, 512
Dense, 1	ReLU	RB, 512	RB, down, 512
	Global sum pooling	ReLU	RB, 512
	Dense, 1	Global sum pooling	ReLU
		Dense, 1	Global sum pooling
			Dense, 1
32×32 resolution	128×128 resolution	256×256 resolution	512×512 resolution
FC, $4 \times 4 \times 256$	FC, $4 \times 4 \times 512$	FC, $4 \times 4 \times 512$	FC, $4 \times 4 \times 512$
RB, up, 256 / GFFM, 256	RB, up, 512 / GFFM, 512	RB, up, 512 / GFFM, 512	RB, up, 512 / GFFM, 512
RB, up, 256 / GFFM, 256	RB, up, 512 / GFFM, 512	RB, up, 512 / GFFM, 512	RB, up, 512 / GFFM, 512
RB, up, 256 / GFFM, 256	RB, up, 256 / GFFM, 256	RB, up, 256 / GFFM, 256	RB, up, 256 / GFFM, 256
BN, ReLU	RB, up, 128	RB, up, 128 / GFFM, 128	RB, up, 128 / GFFM, 128
3×3 conv, Tanh	RB, up, 64 / GFFM, 64	RB, up, 64 / GFFM, 64	RB, up, 64 / GFFM, 64
	BN, ReLU	RB, up, 32 / GFFM, 32	RB, up, 32 / GFFM, 32
	3×3 conv, Tanh	BN, ReLU	RB, up, 16 / GFFM, 16
		3×3 conv, Tanh	BN, ReLU
			3×3 conv, Tanh

it is relevant to keep in the f_i and what information is unnecessary. In contrast, the second term $(1-f_g)\otimes f_r$ indicates that f_i is combined with new information coming from f_s. That means the network can effectively control information flow by adding necessary information derived from the earlier layer.

By using the proposed gating unit, the GFMM produces two different outputs f_m^* and f_a^* which are the refined main and side features, respectively. It is worth noting that the proposed method refines not only f_m but also f_a to store the updated features of the network; f_a acts like a memory cell in the GRU [41]. When producing f_m^*, f_m and f_a are used for f_i and f_s of the gating unit, respectively. In contrast, when making f_a^*, f_m and f_a are used for f_s and f_a of the gating unit, respectively. Consequently, by adding the GFFM between the residual blocks, the generator produces the image while successfully considering the necessary features in the earlier layers.

B. Implementation Details

To reveal the superiority of the proposed method, we conducted extensive experiments using the various standard datasets that are widely used for measuring the GAN performance: CIFAR-10 [30], CIFAR-100 [30], LSUN [31], CelebA-HQ [26], AFHQ [32], and tiny-ImageNet [33], [34] which is a subset of ImageNet [33], consisting of the 200 selected classes. Specifically, among the various classes in the LSUN dataset, we employed the church and tower images in our experiments. The resolutions of the CIFAR-10 and CIFAR-100 datasets are 32×32, whereas we resized the images in LSUN and tiny-ImageNet datasets to 128×128 pixels. In addition, the CelebA-HQ and AFHQ datasets are used for training the networks that produce the 256×256 and 512×512 images. To train the network, we used the hinge-adversarial loss [36]. Since all network parameters in the discriminator and generator including the GFFM can be differentiated, we used the Adam optimizer [43] and set the user parameters, i.e., β_1 and β_2, to 0 and 0.9, respectively.

For training the CIFAR-10 and CIFAR-100 datasets, we set the learning rate as 0.0002 for both the discriminator and generator. The discriminator was updated five times using different mini-batches while the generator is updated once. We set the batch size of the discriminator as 64 and trained the generator for 50 k iterations. Following the previous briefs [10], [17], [18], the generator was trained with a batch size twice as large as when training the discriminator. In other words, the generator and discriminator were trained with 128 and 64 batch sizes, respectively. On the other hand, for training the LSUN and tiny-ImageNet datasets, we employed a two-time scale update rule, called TTUR [44], which set the learning rates of the generator and discriminator to 0.0001 and 0.0004, respectively. Since the TTUR method updates the discriminator a single time when the generator is updated once, we set both batch sizes of the discriminator and the generator to 32. The generator is updated to 300 k iterations on the LSUN dataset and 1 M iterations on the tiny-ImageNet dataset. In addition, for training the CelebA-HQ and AFHQ datasets, we employed the TTUR technique and set both batch sizes of the discriminator and the generator to 16. The generator is updated to 100 k iterations. For all datasets, we decreased the learning rate linearly in the last 50 000 iterations. In addition, For training the CIFAR-10 and CIFAR-100 datasets the SN [10] was only used for the discriminator, whereas the SN was applied to both generator and discriminator when training the LSUN, CelebA-HQ, AFHQ, and tiny-ImageNet datasets.

In our experiments, we built generator and discriminator architectures following a strong baseline in [28], called BigGAN. The detailed discriminator and generator architectures are described in Table II. In the discriminator, the feature maps were down-sampled by utilizing the average pooling after the second convolution layer in the residual block. In the generator, the up-sampling (a nearest-neighbor interpolation) operation was located before the first convolution. In the GFFM, we set $c_g = c_r = c_o$, and c_o in each module is described in Table II. In addition, we matched the spatial size of f_m and f_a using the nearest neighbor interpolation technique after the BN of f_a. When c_a and c_o in the GFFM were different, we used 1×1 convolution having c_o output channels before the BN of f_a to match the number of channels. In the last GFFM, we only used f_m^* to produce the image. On the other hand, to train the network in the cGAN framework, we replaced the BN in the generator with the conditional BN [28], [45] and added the conditional projection layer in the discriminator [18]. More detailed explanations are presented in [18].

TABLE III
COMPARISON OF THE PROPOSED METHOD WITH CONVENTIONAL ONES ON THE CIFAR-10 AND CIFAR-100 DATASETS IN TERMS OF IS AND FID. THE BEST RESULTS ARE IN BOLD

		GAN						cGAN					
Network		SNGAN [18]		BigGAN [28]		Proposed		SNGAN [18]		BigGAN [28]		Proposed	
Dataset		IS↑	FID↓	IS↑	FID↓	IS↑	FID↓	IS↑	FID↓	IS↑	FID↓	IS↑	FID↓
CIFAR-10	trial 1	7.79	13.81	7.70	13.98	8.06	10.92	8.06	10.26	8.09	9.36	8.27	7.96
	trial 2	7.76	13.29	7.80	13.69	8.11	10.96	8.02	10.37	7.97	9.62	8.20	8.20
	trial 3	7.76	13.29	7.87	13.09	8.12	11.12	7.99	10.01	8.02	9.38	8.32	7.83
	Mean	7.77	13.46	7.79	13.58	**8.10**	**11.00**	8.03	10.21	8.03	9.45	**8.26**	**8.00**
	Std	0.02	0.30	0.08	0.45	**0.03**	**0.11**	0.03	0.18	0.06	0.15	**0.06**	**0.19**
CIFAR-100	trial 1	8.08	17.76	7.99	17.45	8.30	14.56	8.62	14.62	8.88	13.17	9.34	10.39
	trial 2	8.08	17.93	7.94	17.72	8.20	15.04	8.62	14.39	8.89	13.20	9.37	10.17
	trial 3	8.05	17.55	7.94	17.57	8.35	14.87	8.80	14.85	8.87	13.01	9.28	10.25
	Mean	8.07	17.75	7.96	17.58	**8.29**	**14.82**	8.68	14.62	8.88	13.13	**9.33**	**10.27**
	Std	0.02	0.19	0.03	0.13	**0.08**	**0.24**	0.10	0.23	0.01	0.10	**0.05**	**0.11**

IV. EXPERIMENTAL RESULTS

A. Evaluation Metric

FID [44] and IS [46] are the most widely used assessments that measure how well the generator synthesizes the image. More specifically, FID, which computes the Wasserstein distance between the distributions of the real and generated samples in the feature space of the Inception model, is defined as follows:

$$F(p,q) = \|\mu_p - \mu_q\|_2^2 + \text{trace}\left(C_p + C_q - 2\left(C_p C_q\right)^{\frac{1}{2}}\right) \quad (4)$$

where $\{\mu_p, C_p\}$ and $\{\mu_q, C_q\}$ are the mean and covariance of the samples with distributions of real and generated images, respectively. A lower FID score indicates that the generated images have better quality. On the other hand, Salimans et al. [46] demonstrated that IS is strongly correlated with the subjective human judgment of image quality. IS is expressed as

$$I = \exp\left(E\left[D_{KL}(p(l|X)||p(l))\right]\right) \quad (5)$$

where l is the class label predicted by the Inception model [47] trained by using the ImageNet dataset [33], and $p(l|X)$ and $p(l)$ represent the conditional class distributions and marginal class distributions, respectively. In contrast with the FID, the better the quality of the generated image, the higher the IS score. In our experiments, we randomly generated 50 000 samples and computed the FID and IS scores using the same number of real images.

B. Experimental Results

To reveal the effectiveness of the proposed method, we conducted preliminary experiments on the image generation task using the CIFAR-10 and CIFAR-100 datasets. In our experiments, we trained the network three times from scratch to show that the performance improvement is not due to the lucky weight initialization. Table III summarizes the comprehensive results that compare the GAN performance between the proposed and conventional methods. On both CIFAR-10 and CIFAR-100 datasets, the proposed method consistently outperformed the current state-of-the-art methods, i e., SNGAN and BigGAN, in terms of FID and IS scores. These results indicate that the proposed method is more effective than the existing residual block-based generators. In addition, in the cGAN framework, the proposed method showed remarkable performance on both datasets and exhibited better performance than its counterpart by a large margin. For instance, on the CIFAR-100 dataset, the proposed method achieved the FID of 10.27, which was about 29.75% and 21.78% better than SNGAN and BigGAN, respectively. Based on these results, we believe that the proposed method can achieve outstanding performance in both GAN and GAN schemes.

TABLE IV
COMPARISON OF THE PROPOSED METHOD WITH CONVENTIONAL ONES ON THE CIFAR-10 AND CIFAR-100 DATASETS IN TERMS OF IS AND FID, WHEN THE NUMBER OF NETWORK PARAMETERS IS THE SAME. BIGGAN† IS A BIGGAN NETWORK BUILT WITH THE DENSE BLOCK

Dataset	Metric	BigGAN (5.68M parameters)	BigGAN† (5.68M parameters)	Proposed (5.68M parameters)
CIFAR-10	IS↑	7.82 ± 0.03	7.90 ± 0.02	**8.10 ± 0.03**
	FID↓	12.14 ± 0.15	11.66 ± 0.23	**11.00 ± 0.11**
CIFAR-100	IS↑	8.19 ± 0.04	8.16 ± 0.02	**8.29 ± 0.08**
	FID↓	16.15 ± 0.16	15.90 ± 0.12	**14.82 ± 0.24**

On the other hand, one might perceive that the performance improvement is due to the increased number of network parameters. We agree that the proposed method requires additional network parameters used in the GFFM. To alleviate this concern, we conducted ablation studies that match the network parameters of the conventional methods with that of the proposed one; we increased the channel of the residual block to match the number of network parameters. Since the proposed method follows BigGAN as the baseline model, we compared the performance of the proposed method with that of BigGAN. As presented in Table IV, even using the same number of network parameters, the proposed method exhibited better performance than the conventional methods. Moreover, since we have already shown that the dense block is more effective than the residual block in Section II-B, we also measured the performance of BigGAN built with the dense block (BigGAN† in Table IV). In our experiments, we adjusted the channel of the dense block to match the number of network parameters. As described in Table IV, the dense block-based BigGAN exhibited better performance than the residual block-based BigGAN but still showed weaker performance than the proposed method. These results contain two important meanings. First, the performance improvement is not only due to the increased number of network parameters. Second, compared with the residual and dense blocks, the proposed method is more effective in improving the GAN performance by propagating the information in the earlier layers to the later ones successfully.

Moreover, we conducted ablation studies to reveal the ability of the auxiliary branch with GFFM. First, to show the effectiveness of the GFFM, we measured the GAN performance with prevalent feature fusion methods, i.e., feature summation and concatenation. The results in Table V show that the GFFM achieves better performance on the CIFAR-10 and CIFAR-100 datasets. In addition, we observed that even using the feature summation or concatenation techniques, the generator with the auxiliary branch outperforms conventional methods. To prove the ability of the auxiliary branch more clearly,

TABLE V

PERFORMANCE EVALUATIONS ON THE CIFAR-10 AND CIFAR-100 DATASETS IN TERMS OF IS AND FID, WHEN USING THE NAIVE FEATURE FUSION MODULES

Dataset	Metric	Summation	Concatenation	GFFM (Proposed)
CIFAR-10	IS↑	7.99 ± 0.08	8.03 ± 0.02	**8.10 ± 0.03**
	FID↓	12.38 ± 0.20	11.59 ± 0.17	**11.00 ± 0.11**
CIFAR-100	IS↑	8.07 ± 0.04	8.18 ± 0.01	**8.29 ± 0.08**
	FID↓	16.98 ± 0.10	16.29 ± 0.09	**14.82 ± 0.24**

TABLE VI

PERFORMANCE EVALUATIONS ON THE CIFAR-10 AND CIFAR-100 DATASETS IN TERMS OF IS AND FID TO VERIFY THE EFFECTIVENESS OF THE AUXILIARY BRANCH

Dataset	Shortcut connection	Auxiliary branch	IS↑	FID↓
CIFAR-10			7.76 ± 0.12	14.36 ± 0.40
	✓		7.79 ± 0.08	13.58 ± 0.45
		✓	7.99 ± 0.08	11.20 ± 0.11
	✓	✓	**8.10 ± 0.10**	**11.00 ± 0.11**
CIFAR-100			7.88 ± 0.06	17.78 ± 0.12
	✓		7.96 ± 0.03	17.58 ± 0.13
		✓	**8.30 ± 0.09**	15.17 ± 0.32
	✓	✓	8.29 ± 0.08	**14.82 ± 0.24**

TABLE VII

COMPARISON OF THE PROPOSED METHOD WITH CONVENTIONAL ONES ON THE LSUN-CHURCH AND LSUN-TOWER DATASETS IN TERMS OF FID. THE BEST RESULTS ARE IN BOLD

Network		SNGAN	BigGAN	Proposed
Dataset		FID↓	FID↓	FID↓
LSUN-church	trial 1	7.83	8.18	5.59
	trial 2	8.16	7.88	5.56
	trial 3	8.22	8.11	5.40
	Mean	8.07	8.06	**5.51**
	Std	0.21	0.15	**0.10**
LSUN-tower	trial 1	12.42	13.89	8.77
	trial 2	12.29	12.38	9.90
	trial 3	12.54	11.98	8.85
	Mean	12.42	12.78	**9.17**
	Std	0.12	0.99	**0.63**

we conducted other ablation studies on how much the auxiliary branch improves the GAN performance. In fact, the proposed method contains two factors that assist the information flow: a shortcut connection in the residual block [27] and the auxiliary branch. To prove the effectiveness of each part, we evaluated the GAN performance of the proposed method, while omitting the shortcut connection or auxiliary branch. As summarized in Table VI, the generator with the auxiliary branch accomplishes better performance than that with the shortcut connection. Furthermore, we observed that the generator achieves the lowest FID score by using the shortcut connection and auxiliary branch together. Based on these results of the ablation studies, we concluded that the GFFM and auxiliary branch improve the GAN performance effectively.

To show the effectiveness of the proposed method when synthesizing images on challenging datasets, we trained the network using the LSUN-church and LSUN-tower datasets. Specifically, on the LSUN-church and -tower datasets, the network is trained by following the experimental setting for GAN. As shown in Table VII, the proposed method exhibited a notable performance compared to the conventional methods by a large margin. That means the proposed method is also effective in generating the image with the complex scene. Moreover, we trained the network using the tiny-ImageNet dataset, following the experimental setting for cGAN. Since we already proved that the performance improvement is not due to the lucky weight initialization in Tables III, IV, and VII, we trained the network a single time from scratch on the tiny-ImageNet dataset. Instead, to show the superiority of the proposed method more reliably, we drew a graph representing the FID and IS scores over training iterations. As shown in Figs. 4 and 5, the proposed method consistently outperformed the conventional methods, i.e., SNGAN and BigGAN, during the training procedure. The final FID and IS scores are summarized in Table VIII. Fig. 6 shows the samples of the generated images. These results show that the proposed method is effective in synthesizing images with complex scenes. Based on these results, we concluded that the proposed method can generate visually pleasing images on challenging datasets.

TABLE VIII

COMPARISON OF THE FINAL FID AND IS SCORES ON THE TINY-IMAGENET DATASET. THE BEST RESULTS ARE IN BOLD

Dataset	Metric	SNGAN	BigGAN	Proposed
tiny-ImageNet	IS↑	20.52	20.23	**25.57**
	FID↓	35.42	35.13	**25.00**

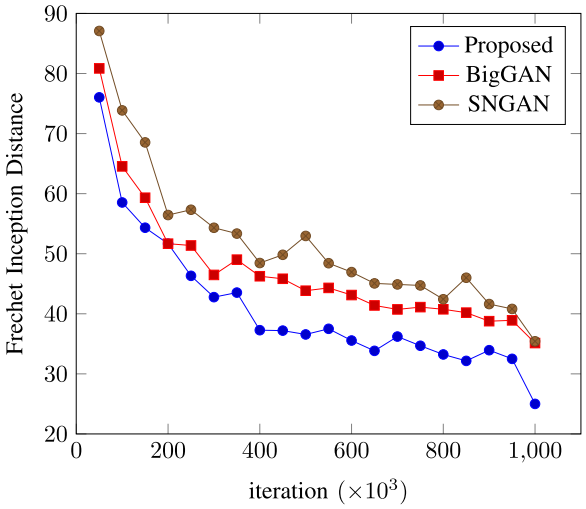

Fig. 4. Comparison of FID scores over training iterations. Blue, red, and yellow lines indicate the FID scores of the proposed method, BigGAN, and SNGAN, respectively.

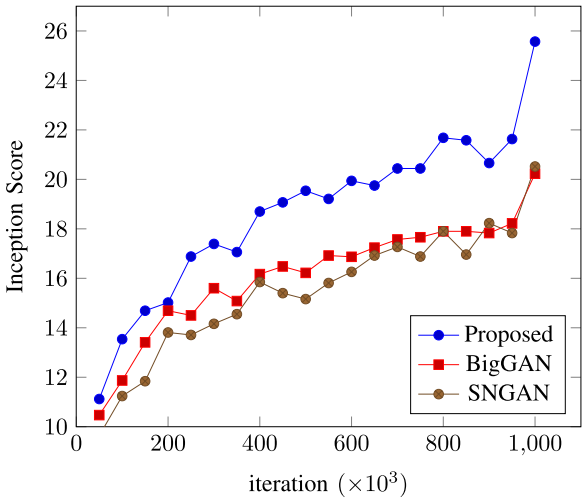

Fig. 5. Comparison of IS scores over training iterations. Blue, red, and yellow lines indicate the IS scores of the proposed method, BigGAN, and SNGAN, respectively.

Fig. 6. Samples of the generated images. (a) Generated images on the LSUN-church dataset. (b) Generated images on the LSUN-tower dataset. (c) Generated images on the tiny-ImageNet dataset.

Furthermore, to show the ability of the proposed method for generating high-resolution images, we performed additional experiments using the CelebA-HQ [26] and AFHQ [32] datasets. In this brief, we trained the networks to produce 256×256 and 512×512 images. However, since the CelebA-HQ and AFHQ datasets are composed of a small number of images, it often suffers from the overfitting problem of the discriminator. To alleviate this problem, we employed PConv to the discriminator [22]. The results in Table IX show that the proposed method is also effective in generating high-resolution images. To prove the generalization ability of the proposed method, we conducted additional experiments by setting StyleGAN2 [48] as the baseline model; we replaced the residual block in StyleGAN2 with the proposed AB block. In our experiments, we set the same number of batch sizes and the training iterations with the BigGAN-based experiments. As reported in Table IX, the proposed method shows lower FID scores than StyleGAN2. In addition, as depicted in Fig. 7, the proposed method with StyleGAN2 produces visually plausible images. Based on these results, we expect that the proposed method can be utilized to generate high-quality images with various baseline models. Indeed, this study does not intend to design an optimal generator and discriminator architecture specialized for the auxiliary branch and GFFM. There might be another network architecture that improves the GAN performance and produces more high-quality images. Instead, this brief focuses on verifying whether it is possible to achieve better GAN performance by simply adding the auxiliary branch and GFFM.

Fig. 7. Generated images with 256×256 and 512×512 resolutions on the Celeb-HQ and AFHQ datasets.

TABLE IX

COMPARISON OF THE PROPOSED METHOD WITH CONVENTIONAL ONES ON THE CELEBA-HQ AND AFHQ DATASETS IN TERMS OF FID

Dataset	Size	BigGAN	Proposed	StyleGAN2	Proposed
CelebA HQ	256	16.27	**11.25**	9.91	**7.90**
	512	27.40	**23.43**	9.18	**7.74**
AFHQ	256	19.02	**16.35**	10.12	**9.13**
	512	22.47	**17.40**	10.73	**8.86**

V. CONCLUSION

This brief has introduced a novel generator architecture with the auxiliary branch and has shown the remarkable performance of the proposed method in various experiments. In addition, we have proposed the GFFM that effectively blends the features in the main and auxiliary branches. The main advantage of the proposed method is that it significantly boosts the GAN performance with a slight number of additional network parameters. Furthermore, this brief has demonstrated the generalization ability of the proposed method in various aspects through high-resolution image generation and extensive experiments. Therefore, we expect that the proposed method is applicable to various GAN-based image generation techniques. However, this brief has aimed at introducing a novel generator architecture specialized to the GAN. We agree that the proposed method works well for the GAN-based image generation technique, but might cause some issues in other networks used for different

applications such as image-to-image translation. There might be another network architecture that shows more general ability than the proposed method. Although this brief only has covered the effectiveness of the proposed method in the field of GAN-based image generation technique, we have shown that the proposed method boosts the GAN performance significantly. In our future work, we plan to further investigate a novel generator architecture that could cover various applications.

REFERENCES

[1] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[2] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," 2016, *arXiv:1605.05396*.

[3] S. Hong, D. Yang, J. Choi, and H. Lee, "Inferring semantic layout for hierarchical text-to-image synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7986–7994.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1125–1134.

[5] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8789–8797.

[6] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.

[7] M.-c. Sagong, Y.-G. Shin, S.-W. Kim, S. Park, and S.-J. Ko, "PEPSI: Fast image inpainting with parallel decoding network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11352–11360.

[8] Y.-G. Shin, M.-C. Sagong, Y.-J. Yeo, S.-W. Kim, and S.-J. Ko, "PEPSI++: Fast and lightweight network for image inpainting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 252–265, Jan. 2021.

[9] Y.-L. Wu, H.-H. Shuai, Z.-R. Tam, and H.-Y. Chiu, "Gradient normalization for generative adversarial networks," 2021, *arXiv:2109.02235*.

[10] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018, *arXiv:1802.05957*.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[12] H. Zhang, Z. Zhang, A. Odena, and H. Lee, "Consistency regularization for generative adversarial networks," 2019, *arXiv:1910.12027*.

[13] B. Wu et al., "Generalization in generative adversarial networks: A novel perspective from privacy protection," 2019, *arXiv:1908.07882*.

[14] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, "Improving the improved training of Wasserstein GANs: A consistency term and its dual effect," 2018, *arXiv:1803.01541*.

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*.

[16] K. Kurach, M. Lučić, and X. Zhai, "A large-scale study on regularization and normalization in GANs," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3581–3590.

[17] S. Park and Y.-G. Shin, "Generative residual block for image generation," *Appl. Intell.*, vol. 52, no. 7, pp. 7808–7817, May 2022.

[18] T. Miyato and M. Koyama, "CGANs with projection discriminator," 2018, *arXiv:1802.05637*.

[19] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7354–7363.

[20] Y.-J. Yeo, Y.-G. Shin, S. Park, and S.-J. Ko, "Simple yet effective way for improving the performance of GAN," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1811–1818, Apr. 2022.

[21] W. Li et al., "DW-GAN: Toward high-fidelity color-tones of GAN-generated images with dynamic weights," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023, doi: 10.1109/TNNLS.2023.3311545.

[22] S. Park, Y.-J. Yeo, and Y.-G. Shin, "PConv: Simple yet effective convolutional layer for generative adversarial network," 2021, *arXiv:2101.10841*.

[23] M.-C. Sagong, Y.-J. Yeo, Y.-G. Shin, and S.-J. Ko, "Conditional convolution projecting latent vectors on condition-specific space," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 1386–1393, Jan. 2022, doi: 10.1109/TNNLS.2022.3172512.

[24] S. Park and Y.-G. Shin, "Generative convolution layer for image generation," 2021, *arXiv:2111.15171*.

[25] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[26] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," 2017, *arXiv:1710.10196*.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[28] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," 2018, *arXiv:1809.11096*.

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4700–4708.

[30] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.

[31] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," 2015, *arXiv:1506.03365*.

[32] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "StarGAN v2: Diverse image synthesis for multiple domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8188–8197.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2009, pp. 248–255.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[35] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2794–2802.

[36] J. Hyun Lim and J. Chul Ye, "Geometric GAN," 2017, *arXiv:1705.02894*.

[37] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[38] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.

[39] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2019, pp. 4401–4410.

[40] C. Yang, Y. Shen, and B. Zhou, "Semantic hierarchy emerges in deep generative representations for scene synthesis," *Int. J. Comput. Vis.*, vol. 129, pp. 1451–1466, Feb. 2021.

[41] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[42] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[44] 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 *Proc. NIPS*, 2017, pp. 6626–6637.

[45] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *Proc. ICLR*, vol. 2, 2017.

[46] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.

[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[48] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8110–8119.