Image generation with self pixel-wise normalization

Yoon-Jae Yeo¹ · Min-Cheol Sagong¹ · Seung Park² · Sung-Jea Ko¹ · Yong-Goo Shin³

Accepted: 13 July 2022 / Published online: 7 September 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Region-adaptive normalization (RAN) methods have been widely used in the generative adversarial network (GAN)-based image-to-image translation technique. However, since these approaches need a mask image to infer pixel-wise affine transformation parameters, they are not applicable to general image generation models having no paired mask images. To resolve this problem, this paper presents a novel normalization method, called self pixel-wise normalization (SPN), which effectively boosts the generative performance by carrying out the pixel-adaptive affine transformation without an external guidance map. In our method, the transforming parameters are derived from a self-latent mask that divides the feature map into foreground and background regions. The visualization of the self-latent masks shows that SPN effectively captures a single object to be generated as the foreground. Since the proposed method produces the self-latent mask without external data, it is easily adaptable to existing generative models. Extensive experiments on various datasets reveal that our SPN significantly improves the performance of image generation technique in terms of Frechet inception distance (FID) and Inception score (IS).

Keywords Generative adversarial networks · Image generation · Normalization · Region-adaptive normalization

1 Introduction

Generative adversarial network (GAN) [9] based on convolutional neural network (CNN) has led a series of breakthroughs for various applications including image-to-image translation [5, 14, 31, 51] and image inpainting [35, 38, 45]. However, due to the instability problem during the training procedure, it is still a challenge to produce high-quality images [37]. Since a goal of GAN is discovering the Nash equilibrium of non-convex game in the high-dimensional parameter space, GAN is substantially more complex and difficult to train compared to networks trained by supervised learning [49]. To address this issue, some papers [4, 16, 48] investigated novel network architectures for discriminator and generator. Although these methods can produce high-resolution images on challenging datasets such as ImageNet [21], they still have the fundamental problem related to the training instability.

Instead of redesigning the network architecture, many studies [10, 19, 26, 28, 32, 34, 43, 49, 50] attempted to penalize the discriminator for alleviating the training instability problem. The spectral normalization (SN) [28] is the most widely practiced normalization technique. In SN, the Lipschitz constraint is imposed by dividing weight matrices of the discriminator with an approximation of their largest singular value. Gulrajani et al. [10] proposed the gradient penalty that regularizes the gradient norm of straight lines, i.e. decision boundaries, between the real and generated samples. Since these normalization or regularization techniques not only mitigate the training instability problem but also effectively improve the GAN performance, most recent works apply those techniques to their applications.

On the other hand, there are only a few attempts to investigate the normalization technique for the generator. Radford et al. [33] proposed the GAN architecture called DCGAN and empirically proved that a batch normalization (BN) [13] is effective for the generator. For a conditional GAN (cGAN) that focuses on producing class-conditional images, Dumoulin et al. [8] introduced a conditional batch normalization (cBN) which performs different affine transformations according to a given condition. Brock et al. [4] slightly modified cBN to infer the transforming parameters from not only the given class but also the
latent vector. In these papers, they reveal that BN and cBN are more effective for the image generation task than other popular normalization techniques such as instance normalization (IN) [40] and layer normalization (LN) [3]. Hence, most existing works adopt BN or cBN for the generator [4, 10, 19, 26, 28, 29, 32, 34, 43, 49, 50].

Recently, some studies introduce region-adaptive normalization (RAN) techniques [15, 24, 31, 46] which perform pixel-wise affine transformations. More specifically, these methods employ the spatially-varying scaling and shifting parameters which are derived from a given mask image such as a semantic segmentation map. Although they exhibit fine performance in the field of the image-to-image translation, there is one major drawback: these approaches assume that the dataset contains reference and mask image pairs [31]. Therefore, the existing RAN methods are not applicable to the general image generation task that does not have paired mask images. For simplicity, in the remainder of this paper, we will regard that GAN represents the image generation task.

Indeed, most current studies [4, 10, 19, 26, 28, 29, 32, 34, 49, 50] train the generative models using standard datasets, e.g. CIFAR-10 [20], CIFAR-100 [20], and ImageNet [7], built for a image classification task. Since the image classification aims at classifying an object, the images of those datasets have only a single object as the foreground. Thus, a generative model trained on those datasets produces the images having the single-attributed foreground and the rest. Based on these observations, we expect that the RAN techniques can be applied to GAN if there is an extra data that discriminates between the foreground and background. However, it is not possible to pre-build the pairs of generated and mask images because we cannot predict which image will be generated.

To resolve this problem, this paper presents a novel normalization method, called self pixel-wise normalization (SPN), which performs pixel-adaptive affine transformation without the external mask image. In the proposed method, SPN produces a self-latent mask that divides the input feature maps into the foreground and background regions. Then, the intrinsic transforming parameters of each region are inferred from the self-latent mask. The proposed method is simple but surprisingly effective for the image generation. To reveal the superiority of the proposed method, we conduct extensive experiments with various datasets including CIFAR-10 [20], CIFAR-100 [20], LSUN [44], and tiny-ImageNet [7, 42]. In addition, we conduct plenty of ablation studies to prove the generalization ability of the proposed method. Quantitative evaluations show that the proposed method significantly improves the performance of both GAN and cGAN in terms of various metrics such as Frechet inception distance (FID) [11] and Inception score (IS) [37].

Key contributions of our paper are summarized as follows:

- For the image generation task, we introduce a new approach to carry out pixel-adaptive affine transformation without the external data. Specifically, we propose a novel normalization method, called SPN, which effectively boosts the generative performance.
- We demonstrate that SPN effectively improves the performance of both GAN and cGAN. For instance, the proposed method significantly improves FID and IS on tiny-ImageNet dataset from 35.13 and 20.23 to 28.31 and 23.35, respectively.
- The proposed method can be easily implemented to the existing state-of-the-art generators without modifying the network architectures.

### 2 Preliminaries

**Generative adversarial network** In the original setting, GAN [9] is composed of the generator $G$ and discriminator $D$. In general, both networks are trained simultaneously but their objectives are contrary: $G$ is trained to generate visually plausible samples, whereas $D$ is optimized to classify real and generated ones. In order to improve the training stability and performance, many studies have been done on reformulation of the objective function. For instance, Mao et al. [25] built the objective function using the least square errors, called least-square GAN (LSGAN), whereas Arjovsky et al. [2] introduced Wasserstein GAN (WGAN) that computes the loss value by measuring the Wasserstein distance between the real and generated samples. Another widely used objective function is a hinge adversarial loss [23]. More specifically, it is formulated as

$$L_D = E_{x \sim P_{data}(x)}[\max(0, 1 - D(x))] + E_{z \sim P_{z}(z)}[\max(0, 1 + D(G(z)))],$$  \hspace{1cm} (1)$$

$$L_G = -E_{z \sim P_{z}(z)}[D(G(z))],$$  \hspace{1cm} (2)$$

where $x$ and $z$ indicate a real image and a latent vector sampled from the data distribution $P_{data}(x)$ and the noise distribution $P_{z}(z)$, respectively. By optimizing both objective functions, $G$ can generate images that resemble the real ones.

On the other hand, cGAN, which aims at producing the class-conditional samples, has also been actively studied [27, 29, 30]. In general, cGAN employs additional conditional information, e.g. class labels or text conditions, in order to control the data generation process. By optimizing the objective functions for cGAN [27, 29] the generator can select the image class to be generated. The
reader is encouraged to review the conventional GAN and cGAN techniques for more details.

**Normalization for GAN** As mentioned in Section 1, most existing works employ BN to their generators instead of using other normalization techniques such as IN [40] and LN [3]. To prove that BN is more suitable for the GAN training, we compare the performance of generative models trained with BN, IN, and LN. As described in Table 1, the generator with BN outperforms those adopting other normalization methods, and Kurach et al. [22] also support this. Based on these observations, we design SPN for incorporating the advantages of BN.

**Region-adaptive normalization** The RAN methods [15, 24, 31, 46] are widely used for the image-to-image translation tasks, e.g., the image inpainting and the semantic image synthesis. Unlike the earlier normalization techniques such as BN and cBN, RAN requires the external data containing information differentiated by each pixel. They generally modulate the normalized feature maps by using a region-specific affine transformation whose parameters are derived from the external data. For example, Park et al. [31] estimated the modulating parameters from the semantic segmentation mask. Jakoel et al. [15] introduced inference-time adaptive normalization (ITAN), an operation that varies spatially according to a user-entered guidance map for spatial control over image-to-image translation such as local generation and semantic attribute transfer. However, these approaches are not applicable to the fundamental image generation task due to the absence of the mask images paired with the generated images. In other words, since the generator produces the images from the input random noise, it is hard to prepare the paired mask image. To overcome the limitation of the conventional RAN techniques, this paper presents a novel form of intrinsic normalization method specialized to the image generation.

### 3 Proposed method

Let $x \in \mathbb{R}^{B \times H \times W \times C}$ be a four-dimensional tensor where $B$, $H$, $W$, and $C$ indicate the size of mini-batch, height, width, and channels, respectively.

|        | BN [13]   | IN [40]  | LN [3]   |
|--------|-----------|----------|----------|
| FID    | 13.46±0.30| 36.19±12.2| 17.71±1.54|
| IS     | 7.77±0.02 | 6.35±0.19 | 7.48±0.12 |

The bold numbers indicate the best performance among the results

### 3.1 Self pixel-wise normalization

Before presenting the proposed method, we briefly introduce BN [13]. In BN, the input feature maps $x$ are normalized in a channel-wise manner and modulated with the learned scaling and shifting parameters. More specifically, the $j$-th channel of output feature maps $y(j) \in \mathbb{R}^{B \times H \times W}$ is computed as follows:

$$y(j) = \gamma_{BN}(j) \left( \frac{x(j) - E[x(j)]}{\sqrt{\text{Var}[x(j)]}} \right) + \beta_{BN}(j)$$

$$= \gamma_{BN}(j) \hat{x}(j) + \beta_{BN}(j),$$

(3)

where $\gamma_{BN}(j)$ and $\beta_{BN}(j)$ indicate scaling and shifting parameters of $j$-th channel, respectively, and $\hat{x}(j) \in \mathbb{R}^{B \times H \times W}$ is a normalized feature map. Since $\gamma_{BN}(j)$ and $\beta_{BN}(j)$ have the same value regardless of the pixel location, it is not available to conduct spatially-varied transformation on $\hat{x}(j)$.

In contrast to BN, we design SPN to vary $\hat{x}(j)$ with respect to the pixel location. Figure 1 illustrates the overall framework of SPN. In the proposed method, $x$ is normalized on a per-channel basis and modulated on a per-pixel basis using the learned scaling and shifting parameters. First, SPN produces self-latent masks $m \in \mathbb{R}^{B \times H \times W \times C}$ which guide the network to infer the pixel-wise transforming parameters. The key to generating $m$ in an unsupervised manner is that the generator synthesizes images composed of two types: foreground and background (see Section 1). That means, if the network produces $m$ that captures two-sided regions well, the region-adaptive scaling and shifting parameters could be derived from $m$.

Based on this hypothesis, SPN draws $m$ by projecting $x$ onto an embedding space and passing through the sigmoid function. However, since the network is trained without a target image, it is not predictable which part of $m$ will be activated. In other words, it is ambiguous whether the active region in $m$ is the foreground. To avoid this issue, we adopt another mask $m^*$, called inverted mask, which is simply obtained by subtracting $m$ from one. Consequently, $m$ and $m^*$ are complementary to represent the two-sided regions. To clearly show the role of $m$, we present an example of visualized masks in Fig. 2. In this example, we train the network on LSUN-church dataset and select some channels in the last SPN layer. Compared to the generated image in the spatial domain, $m$ clearly distinguishes object information even though it is latent data. Namely, without the help of external guidance map, the network builds $m$ which precisely separates the foreground and background well by itself. Therefore, by using $m$ and $m^*$, the scaling and shifting parameters could be estimated so as adaptive for each region.

To this end, we design an affine parameter estimation module that infers the pixel-wise transforming parameters. Since $m$ and $m^*$ have different activation maps for each
Fig. 1 Details of SPN. Given the intermediate feature maps, the self-latent masks are produced through few layers for each channel. Based on each mask, the scaling and shifting parameters, i.e. $\gamma$ and $\beta$, are estimated for each pixel. Since these parameters are distinct in all spatial and dimensional positions, the proposed method performs the pixel-adaptive affine transformation channel, we derive the scaling and shifting parameters for each channel independently. That means, we employ the depth-wise convolution [12] to estimate the transforming parameters from the $j$-th channel of $m$ and $m^*$. Specifically, this procedure can be formulated as follows:

$$\gamma(j) = m(j) \otimes w_{\gamma}^1(j) + m^*(j) \otimes w_{\gamma}^2(j), \quad (4)$$

$$\beta(j) = m(j) \otimes w_{\beta}^1(j) + m^*(j) \otimes w_{\beta}^2(j), \quad (5)$$

where $w_{\gamma}^1(j)$ and $w_{\beta}^1(j)$ represent different convolutional weights for the $j$-th channel, and $\otimes$ is the convolution operation. In addition, $\gamma(j)$ and $\beta(j)$ indicate the pixel-wise scaling and shifting parameters for the $j$-th channel, respectively. Note that since $\gamma(j)$ and $\beta(j)$ are inferred without affecting to other channels, the network can learn the appropriate parameters by considering the local characteristics in each channel. Figure 3 shows how to derive the scaling and shifting parameters from the each channel of $m$ and $m^*$. As described in Fig. 3, the proposed method could predict the adaptive $\gamma(j)$ and $\beta(j)$ values for each pixel. Consequently, SPN is formulated as

$$\hat{x}(j) = \gamma(j) \hat{x}(j) + \beta(j). \quad (6)$$

This equation exhibits that $\hat{x}(j)$ is scaled and shifted by the pixel-wise affine transformation, which allows the generator to learn the region-specific features.

3.2 Conditional SPN

In the field of cGAN [4, 27], cBN [8], which produces the different affine transformation parameters according to the given condition, is widely used. Recently, Brock et al. [4] added the output of the fully-connected layer, which receives the noise vector as input, to allow the latent space to directly affect the affine transformation. Specifically, this approach is mathematically equivalent to adding a bias term

Fig. 2 Visualization of the self-latent masks. (a) Self latent masks $m$. (b) Inverted masks $m^*$ corresponding to the above $m$. Generated image
derived from the noise vector to the scaling and shifting parameters in cBN. Since it shows better performance than the conventional cBN, we design the conditional SPN based on the method in [4].

In order to estimate the conditional $\gamma(j)$ and $\beta(j)$, i.e. $\gamma_c(j)$ and $\beta_c(j)$, we modify the two components of SPN: the procedure of building the self-latent mask and the convolution operation in the affine parameter estimation module. First, when producing the self-latent mask, we replace BN with cBN to embed the condition-specific information. That means, we attempt to separately predict the foreground and background regions for each condition. Second, we employ a conditional convolution (cConv) [36] in the affine parameter estimation module. To provide the conditional information, cConv generates the condition-specialized weights from an embedding of class vector. Thus, owing to the conditional information in cConv, we could produce the $\gamma_c(j)$ and $\beta_c(j)$ which have different values according to the given condition. In addition, like the existing method [4], we add the bias term derived from the noise vector. Figure 4 shows how the proposed method for cGAN estimates the conditional scaling and shifting parameters from each channel of $m$ and $m^*$.

4 Experiments

4.1 Implementation details

Datasets To show the superiority of the proposed method, we conduct extensive experiments on the various datasets: CIFAR-10 [20], CIFAR-100 [20], LSUN [44], and tiny-ImageNet [42], which is a subset of ImageNet [7], consisting of the 200 selected classes. Specifically, among the various classes in LSUN dataset, we employ the church, tower, and car images to our experiments. The resolutions of CIFAR-10 and CIFAR-100 datasets are $32 \times 32$, whereas images of LSUN and tiny-ImageNet datasets are resized to $128 \times 128$ pixels. We use the hinge-version loss as the objective function in all experiments, except for the ablation studies in Table 7. In addition, we employ the Adam optimizer [18] and set the user parameters of Adam optimizer, i.e. $\beta_1$ and $\beta_2$, to 0 and 0.9, respectively. Exceptionally, we set $\beta_2$ 0.99 when training the LSUN dataset.

For training CIFAR-10 and CIFAR-100 datasets, we set the learning rate as $2e^{-4}$, and the discriminator was updated 5 times using different mini-batches when the generator is updated once. Also, we set a batch size of the discriminator as 64 and trained the generator for 50k iterations. In our experiments, following the previous papers [28, 29, 32], the generator is trained with a batch size twice as large as when training the discriminator. That means the generator and discriminator are trained with 128 and 64 batch size, respectively. In contrast, for training LSUN and tiny-ImageNet datasets, we employ a two-time scale update rule (TTUR) [11] where the learning rates of the generator and discriminator are set to $1e^{-4}$ and $4e^{-4}$, respectively. In the TTUR technique, the discriminator is updated a single time when the generator is updated once. We set batch sizes of the discriminator and the generator to 32. The network is trained for 300k iterations on the LSUN dataset, and
1M iterations on the tiny-ImageNet dataset. We reduce the learning rate linearly over the last 50k iterations for all datasets. All models are trained on a single RTX 3090 GPU.

**Network architecture** To measure the effectiveness of the proposed method, like the state-of-the-art studies [4, 29, 32, 47], we design the generator and discriminator following the strong baselines, i.e. SNGAN and BigGAN [4, 28]. More specifically, we adopt the generator and discriminator architectures constructing with multiple residual blocks as our baseline models. To train the networks, we replace the conventional normalization techniques of the residual block in the generator with the proposed method. Figure 5 shows the detailed architectures of the residual blocks in the conventional and proposed methods.

![Fig. 4](image)

**Fig. 4** The illustration of the conditional affine parameter estimation module which produces $\gamma_c(j)$ and $\beta_c(j)$ using $m(j)$ and $m^*(j)$

![Fig. 5](image)

**Fig. 5** The illustration of the residual blocks. (a) A residual block in SNGAN. (b) A residual block (condition) in SNGAN. (c) A residual block (condition) in BigGAN. (d) A residual block in the proposed method. (e) A residual block (condition) in the proposed method
For training CIFAR-10 and CIFAR-100 datasets the spectral normalization (SN) [28] is only used for the discriminator, whereas the SN is applied to both generator and discriminator for training LSUN and tiny-ImageNet datasets. In the discriminator, the feature maps are downsampled by utilizing the average-pooling after the second convolution. In the generator, the up-sampling (a nearest-neighbor interpolation) operation is located before the first convolution. Descriptions of the network architectures for the generator and discriminator are described in Table 2. When training on LSUN dataset, both leaky ReLU activation and minibatch standard deviation layers are used for the discriminator as in StyleGAN V2 [17]. To train the network in the conditional GAN (cGAN) framework, following the most representative cGAN scheme, we add the conditional projection layer in the discriminator [29].

**Evaluation metrics** We employ the most popular assessments, FID [11] and IS [37], which evaluate how realistic the generated image is. In particular, the FID computes the Wasserstein-2 distance between the distributions of the generated and real samples in the feature space of the Inception model [39]. The generated samples with better quality have lower FID. On the other hand, the IS measures the KL divergence between the conditional and marginal class distributions. In contrast with the FID, the better the quality of the generated image, the higher the IS. In our experiments, we randomly generate 50,000 samples and compute the FID.

| Generator | Discriminator |
|-----------|---------------|
| $z \in \mathbb{R}^{128} \sim N(0, I)$ | RGB image |
| FC, $4 \times 4 \times 256$ | ResBlock, down, 128 |
| ResBlock*, up, 256 | ResBlock, down, 128 |
| ResBlock*, up, 256 | ResBlock, 128 |
| ResBlock*, up, 256 | ResBlock, 128 |
| BN, ReLU | ReLU |
| $3 \times 3$ Conv, Tanh | Global sum pooling |

| Network architectures of the generator and discriminator for $32 \times 32$ (top) and $128 \times 128$ (bottom) images |
|-----------------|-----------------|
| $z \in \mathbb{R}^{128} \sim N(0, I)$ | RGB image |
| FC, $4 \times 4 \times 512$ | ResBlock, down, 64 |
| ResBlock, up, 512 | ResBlock, down, 128 |
| ResBlock, up, 512 | ResBlock, down, 256 |
| ResBlock*, up, 256 | ResBlock, down, 128 |
| ResBlock*, up, 256 | ResBlock, 512 |
| ResBlock*, up, 64 | ResBlock, 512 |
| BN, ReLU | ReLU |
| $3 \times 3$ Conv, Tanh | Global sum pooling |

Table 2
Fig. 6 Samples of the generated images using the proposed method on CIFAR-10 and CIFAR-100 datasets

and IS using the same number of the real images. We also adopt additional measures, i.e. kernel Inception distance (KID), precision and recall (P/R), to ensure the reliability of the quantitative results.

4.2 Experimental results

Following the state-of-the-art studies [4, 29, 47], we employ a strong baseline called spectrally normalized GAN (SNGAN) [28], which is widely used to this day; we only replace the normalization in the generator with the proposed SPN and do not alter the architecture of the discriminator. As the comparison algorithm, we further adopt the recent study for the image generation, i.e. StyleGAN V2 [17]. For cGAN, specifically, BigGAN [4] is a SNGAN-like structure in which noise is additionally input to the residual block. Thus, in the cGAN scheme, we compare the performance of the proposed method with those of SNGAN and BigGAN.

To demonstrate the effectiveness of the proposed SPN, we conduct the preliminary experiments of the image generation task on CIFAR-10 and CIFAR-100 datasets. Table 3 summarizes the comprehensive results obtained by applying the conventional and proposed methods, and the bold numbers indicate the best performance among the results. As shown in Table 3, on both CIFAR-10 and CIFAR-100 datasets, SPN consistently achieves the better performance than its counterpart in terms of FID and IS. These results reveal that using SPN is more effective than utilizing BN in the generator. In addition, in the cGAN framework, the proposed method outperforms the current state-of-the-art methods, i.e. SNGAN and BigGAN, by a large margin in all the datasets. For instance, on CIFAR-100 dataset, the proposed method achieves the FID of 10.34.

Table 4 Comparison of GAN performance on LSUN dataset

| Dataset | SNGAN [28] | StyleGAN V2 [17] | Proposed |
|---------|------------|------------------|----------|
| LSUN    | 6.98       | 6.68             | 5.93     |
| -church | 0.0043     | 0.0040           | 0.0030   |
| [44]    | 0.63/0.36  | 0.58/0.34        | 0.63/0.36|
| LSUN    | 10.09      | 8.60             | 7.23     |
| -tower  | 0.0048     | 0.0041           | 0.0034   |
| [44]    | 0.58/0.23  | 0.58/0.26        | 0.58/0.33|
| LSUN    | 10.46      | 8.05             | 6.19     |
| -car    | 0.0042     | 0.0028           | 0.0021   |
| [44]    | 0.61/0.09  | 0.64/0.19        | 0.69/0.29|

The bold numbers indicate the best performance among the results.
which is about 29.27% and 21.25% better than SNGAN and BigGAN, respectively. Figure 6 shows the qualitative results for these datasets.

To ensure the ability of SPN when producing images in challenging datasets, we employ some categories of the LSUN dataset: church, tower, and car. As shown in Table 4, the proposed method effectively improves the generative performance in terms of the KID and P/R as well as FID compared to SNGAN and StyleGAN V2. Furthermore, to evaluate the effectiveness of conditional SPN, we train the network using the tiny-ImageNet dataset. In our experiments, to show the effectiveness of conditional SPN more reliably, we further present the FID and IS curves exhibiting the performance growth on tiny-ImageNet over the training iteration in Fig. 7. As we can see, the proposed method consistently outperforms the conventional methods, i.e. SNGAN and BigGAN, during the training procedure. The final FID and IS values are summarized in Table 5. Based on these results, we conclude that the proposed method is effective to significantly improve the performance of cGAN as well as GAN. On the other hand, Fig. 8 shows the samples of the generated images. As we can see, the proposed method is effective to synthesize the images with complex scenes. These overall results indicate that SPN and conditional SPN can generate visually pleasing images on challenging datasets.

4.3 Ablation studies

Indeed, SPN employs slightly more network parameters to build the self-latent mask and to predict the affine transformation parameters. Thus, for SPN to be practically used, it needs to achieve a fine trade-off between improved performance and increased computational cost. To explain the efficiency of SPN clearly, we compare the number of network parameters and floating operations (FLOPs) of the conventional and proposed methods. As shown in Table 6, the proposed method replacing 6 BNs of SNGAN with SPNs uses additional parameters of 0.05M and FLOPs of 0.20B; the proposed method marginally increases the computational complexity. One may anticipate that the increased number of computational cost results in improving the performance. To alleviate this issue, we conduct ablation study that equalizes the computational cost of the generator. More specifically, we increase the number of output channels in the generator of SNGAN so as to match the computational requirement. As described in Table 6, this variation does not contribute to any noticeable performance improvement. Thus, we believe that the performance improvement is caused by the spatially varied transformation, not by the additional computational cost. In order to show the generalization ability of SPN, we conduct some ablation studies of the detailed components in the proposed method. Table 7 lists the performance of variations of the proposed SPN. First, we vary the convolutional kernel size acting on $m$ and $m^*$. It hurts the performance to use a kernel size of $1 \times 1$ since it cannot utilize the local context

| Model       | tiny-ImageNet [7, 42] |
|-------------|------------------------|
|             | FID   | IS    |
| SNGAN [28]  | 35.42 | 20.52 |
| BigGAN [4]  | 35.13 | 20.23 |
| Proposed    | **28.31** | **23.35** |

The bold numbers indicate the best performance among the results.
information of the masks. Meanwhile, a kernel size of $5 \times 5$ shows similar performance to the $3 \times 3$ kernel. Thus, in the rest of our experiments, we adopt the kernel size of $3 \times 3$ for computational efficiency. On the other hand, to show the validity of producing an independent mask for each channel, we measure the performance of the model trained with $m$ having a single channel. As summarized in Table 7, the single channel mask degrades performance since it has insufficient variations to derive the proper transformation parameters for each channel, i.e. $\gamma(c(j))$ and $\beta(c(j))$. In addition, instead of using the depth-wise convolution, we apply the standard convolution to $m$ having $c$ channels to estimate the transforming parameters. However, this model fails to train since there are too many parameters in the normalization layer; it is hard to train stably because of an imbalance between the generator and discriminator.

In the conditional SPN (cSPN), we employ the noise vector to produce the bias term for $\gamma(c(j))$ and $\beta(c(j))$. To show the effectiveness of the bias term, we conduct ablation studies that compare the performance of the network trained with/without the bias term. The experimental results are summarized in Table 8. As we can see, the bias term slight improves the performance of the proposed cSPN on both CIFAR-10 and CIFAR-100 datasets. These results indicate that the bias term derived from the latent space can encourage the performance improvement. It is worth noting that the proposed method without the bias term still outperforms the conventional methods, i.e. cBN and ccBN. That means even someone builds the generator that is difficult to provide the bias term to cSPN, the network

Table 6 When adopting BN or SPN, the generative performance on CIFAR-10 dataset and number of network parameters and FLOPs

|           | SNGAN   | SNGAN†  | Proposed |
|-----------|---------|---------|----------|
| FID       | 13.46±0.30 | 13.61±0.24 | 12.16±0.16 |
| IS        | 7.77±0.02  | 7.74±0.05  | 7.93±0.02  |
| #Param.   | 4.07M    | 4.53M    | 4.52M    |
| #FLOPs    | 1.59B    | 1.79B    | 1.79B    |

† means a channel-increased model so as to match the computational cost of the proposed method. Here, M/B indicates million/billion (10^6/10^9), respectively.

The bold numbers indicate the best performance among the results.

Table 7 Performance of the proposed method with different configurations on CIFAR-10 dataset

| Kernel size | #Ch. of $m$ | FID     | IS     |
|-------------|-------------|---------|--------|
| 1 × 1       | $c$         | 12.26±0.29 | 7.82±0.01 |
| 3 × 3       | $c$         | 12.16±0.16 | 7.93±0.02 |
| 5 × 5       | $c$         | 12.26±0.35 | 7.99±0.03 |
| 3 × 3       | 1           | 12.76±0.28 | 7.88±0.05 |
| 3 × 3       | $c^*$       | 67.09±8.69 | 5.57±0.09 |

We change the kernel size of the convolution in the affine parameter estimation module and the number of channels of $m$. Also, $*$ indicates the case of employing the standard convolution instead of the depth-wise convolution.

The bold numbers indicate the best performance among the results.
Table 8 Performance evaluation of the conventional and proposed methods with/without the bias term on CIFAR-10 (top) and CIFAR-100 (bottom) datasets

| Method      | Bias | FID     | IS      |
|-------------|------|---------|---------|
| cBN         |      | 10.21±0.18 | 8.03±0.03 |
| ccBN        | ✓    | 9.45±0.15  | 8.03±0.06  |
| cSPN (ours) | ✓    | 7.72±0.18  | 8.35±0.06  |
| cBN         |      | 14.62±0.23 | 8.68±0.10  |
| ccBN        | ✓    | 13.13±0.10 | 8.88±0.01  |
| cSPN (ours) | ✓    | 11.06±0.19 | 9.28±0.06  |
| cSPN (ours) | ✓    | 10.34±0.40 | 9.37±0.10  |

The bold and underlined numbers indicate the best and the second-best performance among the results, respectively.

Table 9 Performance evaluation in different loss settings on CIFAR-10 dataset

| Loss function | Metric | SNGAN | Proposed |
|---------------|--------|-------|----------|
| CE [9]        | FID    | 17.41±0.64 | 15.13±1.25 |
|               | IS     | 7.52±0.02  | 7.65±0.11  |
| LSGAN [25]    | FID    | 20.57±0.83 | 18.74±0.45 |
|               | IS     | 7.17±0.02  | 7.42±0.04  |

The bold numbers indicate the best performance among the results.

To that trained without SA. In contrast, the proposed method slightly improves the generative performance when using SA. Based on these results, we expect that further improvement can be achieved by designing a new SA technique fitting well with the proposed method.

5 Analysis and discussion

Why does SPN work better? A short answer to this question is that SPN is better to provide the region-specific features than the conventional normalization methods. Indeed, while BN [13] is an essential piece in almost all generators, it cannot embed the spatially varied information since it employs the same scaling and shifting parameters for all pixels. In other words, when using BN, the convolutional layer should learn the region-specific features itself. However, this work is quite taxing on the convolutional layer due to the absence of the external guidance. In contrast, since SPN allows the intrinsic pixel-wise affine transformation, the proposed model could produce the spatially adaptive features in the normalization procedure. That means it can offload the burden of the convolutional layer. Therefore, SPN and convolutional layer have a synergistic effect to derive the features appropriate for

```python
def self_pixelwise_normalization(x):
    # Input feature maps 'x' with shape [B, H, W, C]
    # Produce self-latent maps 'm'
    m = Activation('relu')(BatchNormalization(center=True, scale=True)(x))
    m = Activation('sigmoid')(Conv2D(1, kernel_size=1)(m))
    # Estimate the scaling and shifting parameters 'g' and 'b'
    g = DepthwiseConv2D(kernel_size=3)(m) + DepthwiseConv2D(kernel_size=3)(1 - m)
    b = DepthwiseConv2D(kernel_size=3)(m) + DepthwiseConv2D(kernel_size=3)(1 - m)
    # Batch normalization without affine transformation
    x_lat = BatchNormalization(center=True, scale=True)(x)
    # Pixel-wise affine transformation
    return (1 + g) * x_lat + b
```

Fig. 9 Python code of the proposed SPN. This is implemented in the Tensorflow and Keras API.

The bold numbers indicate the best performance among the results.

Furthermore, we conduct additional experiments in which the networks are trained with two different objective functions: the function based on the cross-entropy (CE) theorem and the function proposed in the least square GAN (LSGAN) paper [25]. As shown in Table 9, the proposed method exhibits the performance improvement steadily even with various loss functions. These results reveal that the proposed method can be effectively applied to the conventional GAN without considering experimental settings such as the adversarial loss function.

One might perceive that the proposed method is similar to the technique that directly feeds spatial attention to the feature map. However, while the existing spatial attention methods are additionally applied after the convolutional layer, SPN is an improved normalization algorithm. Therefore, the proposed method is a differentiated technique that can be used together with the spatial attention method. To prove this claim, we conduct experiments that train the network with/without the well-known spatial attention (SA) technique [41]. As presented in Table 10, the generator trained with SA shows analogous performance to that trained without SA. In contrast, the proposed method slightly improves the generative performance when using SA. Based on these results, we expect that further improvement can be achieved by designing a new SA technique fitting well with the proposed method.

Table 10 Performance comparison of the conventional and proposed methods with/without the spatial attention (SA) technique on CIFAR-10 dataset

| Model       | SA [41] | FID     | IS      |
|-------------|---------|---------|---------|
| SNGAN       | ✓       | 13.46±0.30 | 7.77±0.02 |
| Proposed    | ✓       | 12.16±0.16 | 7.93±0.02 |

The bold numbers indicate the best performance among the results.

would achieve the better performance than the existing normalization methods.

The analysis and discussion section concludes with a short summary of why SPN works better than conventional methods, highlighting the synergistic effect between SPN and convolutional layers in producing region-specific features.

The Python code snippet provided demonstrates the implementation of the proposed SPN, showcasing the use of Tensorflow and Keras API.
each region, which can remarkably improve the generative performance. We present the simply abbreviated code for SPN on the Tensorflow [1] platform. The code using the Keras API [6] is presented in Fig. 9.

Furthermore, we theoretically analyze the proposed method. In fact, SPN is related to and generalized to several existing normalization methods. First, when replacing $m(j)$ with the spatially-invariant constant value, we arrive at the form of BN since all values in $\gamma(j)$ or $\beta(j)$ are same. Similarly, we can make the form of LN [3] by replacing $m$ with the scalar value. Unlike the previous normalization methods in which some or all of $m$ is uniform, the proposed SPN generates the spatially adaptive $m$ for each input feature map. Again, $m$ is an internally produced self-latent mask for the feature itself. Hence, the proposed method is more suitable for the image generation.

Limitations Despite the significant improvements, SPN still faces with one confusion: the proposed method is designed with an assumption that the generated images contain a single object and background. That means we do not consider applying SPN to the image-to-image task that synthesizes the images having multiple objects with different classes. Thus, we believe that future investigation should be required to employ SPN to the image-to-image translation technique.

6 Conclusion

In this paper, we introduce a novel normalization technique, called SPN, adopted for the generator of both GAN and cGAN schemes. The proposed method performs the pixel-adaptive affine transformation with only feature-intrinsic data. Without a complex modification, our technique is simply applied to the existing models by replacing the conventional normalization. We validate the superiority of SPN with visualization of its self-latent masks and the extensive experimental results. In addition, we further investigate the proposed method in its broad aspects with ablation studies. Therefore, it is expected that our method is applicable to various GAN-based applications.

Acknowledgments This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2022R1G1A1004001).

References

1. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kadlec M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. http://tensorflow.org/Softwareavailablefromtensorfloworg. Accessed 9 Nov 2015
2. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein GAN. arXiv:1701.07875
3. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. arXiv:1607.06450
4. Brock A, Donahue J, Simonyan K (2018) Large scale GAN training for high fidelity natural image synthesis. arXiv:1809.11096
5. Choi Y, Choi M, Kim M, Ha JW, Kim S, Choo J (2018) StarGAN: unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8789–8797
6. Chollet F et al (2015) Keras. https://github.com/keras. Accessed 27 March 2015
7. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee, pp 248–255
8. Dumoulin V, Shlens J, Kudlur M (2017) A learned representation for artistic style. Proc of ICLR, vol 2
9. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
10. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved training of Wasserstein GANs. In: Advances in neural information processing systems, pp 5767–5777
11. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017) GANS trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in neural information processing systems, pp 6626–6637
12. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
13. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. PMLR, pp 448–456
14. Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1125–1134
15. Jakoel K, Efraim L, Shaham TR (2022) GANS spatial control via inference-time adaptive normalization. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 2160–2169
16. Karras T, Aila T, Laine S, Lehtinen J (2017) Progressive growing of GANs for improved quality, stability, and variation. arXiv:1710.10196
17. Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T (2020) Analyzing and improving the image quality of styleGAN. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8110–8119
18. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
19. Kodali N, Abernethy J, Hays J, Kira Z (2017) On convergence and stability of GANs. arXiv:1705.07215
20. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images
21. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
22. Kurach K, Lučić M, Zhai X, Michalski M, Gelly S (2019) A large-scale study on regularization and normalization in GANs. In: International conference on machine learning. PMLR, pp 3581–3590

23. Lim JH, Ye JC (2017) Geometric GAN. arXiv:1705.02894

24. Ling J, Xue H, Song L, Xie R, Gu X (2021) Region-aware adaptive instance normalization for image harmonization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9361–9370

25. Mao X, Li Q, Xie H, Lau Ry, Wang Z, Paul Smolley S (2017) Least squares generative adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp 2794–2802

26. Mescheder L, Geiger A, Nowozin S (2018) Which training methods for GANs do actually converge?. arXiv:1801.04406

27. Mirza M, Osindero S (2014) Conditional generative adversarial nets. arXiv:1411.1784

28. Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for generative adversarial networks. arXiv:1802.05957

29. Miyato T, Koyama M (2018) cGANs with projection discriminator. arXiv:1802.05637

30. Odena A, Olah C, Shlens J (2017) Conditional image synthesis with auxiliary classifier GANs. In: Proceedings of the 34th international conference on machine learning. JMLR. org. vol 70, pp 2642–2651

31. Park T, Liu MY, Wang TC, Zhu JY (2019) Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2337–2346

32. Park S, Yeo YJ, Shin YG (2021) Generative adversarial network using perturbed-convolutions. arXiv:2101.10841

33. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434

34. Roth K, Lucchi A, Nowozin S, Hofmann T (2017) Stabilizing training of generative adversarial networks through regularization. In: Advances in neural information processing systems, pp 2018–2028

35. Sagong MC, Shin YG, Kim SW, Park S, Ko SJ (2019) PEPSI: fast image inpainting with parallel decoding network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 11360–11368

36. Sagong MC, Yeo YJ, Shin YG, Ko SJ (2022) Conditional convolution projecting latent vectors on condition-specific space. IEEE Trans Neural Netw Learn Syst

37. Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training GANs. In: Advances in neural information processing systems, pp 2234–2242

38. Shin YG, Sagong MC, Yeo YJ, Kim SW, Ko SJ (2020) PEPSI++: fast and lightweight network for image inpainting. IEEE Trans Neural Netw Learn Syst

39. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826

40. Ulyanov D, Vedaldi A, Lempitsky V (2016) Instance normalization: the missing ingredient for fast stylization. arXiv:1607.08022

41. Wang S, Park J, Lee JY, Kweon IS (2018) CBAM: convolutional block attention module. In: Proceedings of the European conference on computer vision (ECCV), pp 3–19

42. Yao L, Miller J (2015) Tiny ImageNet classification with convolutional neural networks. CS 231N 2(5):8

43. Yeo YJ, Shin YG, Park S, Ko SJ (2021) Simple yet effective way for improving the performance of GAN. IEEE Trans Neural Netw Learn Syst

44. Yu F, Seff A, Zhang Y, Song S, Funkhouser T, Xiao J (2015) LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. arXiv:1506.03365

45. Yu J, Lin Z, Yang J, Shen X, Lu X, Huang TS (2018) Free-form image inpainting with gated convolution. arXiv:1806.03589

46. Yu T, Guo Z, Jin X, Wu S, Chen Z, Li W, Zhang Z, Liu S (2020) Region normalization for image inpainting. In: Proceedings of the AAAI conference on artificial intelligence. vol 34, pp 12733–12740

47. Zhang H, Goodfellow I, Metaxas D, Odena A (2019) Self-attention generative adversarial networks. In: International conference on machine learning, pp 7354–7363

48. Zhang H, Xu T, Li H, Zhang S, Wang X, Metaxas DN (2018) StackGAN++: realistic image synthesis with stacked generative adversarial networks. IEEE Trans Pattern Anal Mach Intell 41(8):1947–1962

49. Zhang H, Zhang Z, Odena A, Lee H (2019) Consistency regularization for generative adversarial networks. arXiv:1910.12027

50. Zhou B, Krähenbühl P (2018) Don’t let your discriminator be fooled. In: International conference on learning representations

51. Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp 2223–2232

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Yoon-Jae Yeo received his B.S. degree in Electrical Engineering from Korea University in 2017. He is currently pursuing his Ph.D. degree in Electrical Engineering at Korea University. His research interests are in the areas of image processing, computer vision, and deep learning.

Min-Cheol Sagong received his B.S. degree in Electrical Engineering from Korea University in 2017. He is currently pursuing his Ph.D. degree in Electrical Engineering at Korea University. His research interests are in the areas of deep neural network based computer vision, and artificial intelligence.
Seung Park received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, South Korea, in 2013 and 2020, respectively. He is currently a Clinical Assistant Professor in Chungbuk National University Hospital. His current research interests include computer vision and image processing.

Sung-Jea Ko (M’88–SM’97–F’12) received his Ph.D. degree in 1988 and his M.S. degree in 1986, both in Electrical and Computer Engineering, from State University of New York at Buffalo, and his B.S. degree in Electronic Engineering at Korea University in 1980. In 1992, he joined the Department of Electronic Engineering at Korea University where he is currently a Professor. From 1988 to 1992, he was an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Michigan-Dearborn. He has published over 210 international journal articles. He also holds over 60 registered patents in fields such as video signal processing and computer vision.

Prof. Ko received the best paper award from the IEEE Asia Pacific Conference on Circuits and Systems (1996), the LG Research Award (1999), and both the technical achievement award (2012) and the Chester Sall award from the IEEE Consumer Electronics Society (2017). He was the President of the IEIE in 2013 and the Vice-President of the IEEE CE Society from 2013 to 2016. He is a member of the National Academy of Engineering of Korea. He is a member of the editorial board of the IEEE Transactions on Consumer Electronics.

Yong-Goo Shin received the B.S. and Ph.D. degrees in electronics engineering from Korea University, Seoul, South Korea, in 2014 and 2020, respectively. He is currently an Assistant Professor with the Division of Smart Interdisciplinary Engineering, Hannam University. He has published over 10 international journal articles in fields such as image processing, computer vision, and deep learning.
Affiliations

Yoon-Jae Yeo\textsuperscript{1} · Min-Cheol Sagong\textsuperscript{1} · Seung Park\textsuperscript{2} · Sung-Jea Ko\textsuperscript{1} · Yong-Goo Shin\textsuperscript{3}

Yoon-Jae Yeo
yjyeo@dali.korea.ac.kr

Min-Cheol Sagong
mcsagong@dali.korea.ac.kr

Seung Park
spark.cbnuh@gmail.com

Sung-Jea Ko
sjko@korea.ac.kr

\textsuperscript{1} School of Electrical Engineering Department, Korea University, Anam-dong, Sungbuk-gu, Seoul, 136-713, Republic of Korea

\textsuperscript{2} Biomedical Engineering, Chungbuk National University Hospital, 776, Seowon-gu, Cheongju-si, Chungcheongbuk-do, Republic of Korea

\textsuperscript{3} Department of Artificial Intelligence, Hannam University, Daejeon, 34430, Republic of Korea