Research Article

CPGAN: An Efficient Architecture Designing for Text-to-Image Generative Adversarial Networks Based on Canonical Polyadic Decomposition

Ruixin Ma and Junying Lou

School of Software, Dalian University of Technology, Dalian 116024, China

Correspondence should be addressed to Ruixin Ma; maruixin@dlut.edu.cn

Received 20 January 2021; Revised 24 February 2021; Accepted 11 March 2021; Published 2 April 2021

Academic Editor: Liang Zou

Copyright © 2021 Ruixin Ma and Junying Lou. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Text-to-image synthesis is an important and challenging application of computer vision. Many interesting and meaningful text-to-image synthesis models have been put forward. However, most of the works pay attention to the quality of synthesis images, but rarely consider the size of these models. Large models contain many parameters and high delay, which makes it difficult to be deployed on mobile applications. To solve this problem, we propose an efficient architecture CPGAN for text-to-image generative adversarial networks (GAN) based on canonical polyadic decomposition (CPD). It is a general method to design the lightweight architecture of text-to-image GAN. To improve the stability of CPGAN, we introduce conditioning augmentation and the idea of autoencoder during the training process. Experimental results prove that our architecture CPGAN can maintain the quality of generated images and reduce at least 20% parameters and flops.

1. Introduction

Text-to-image synthesis is a challenging cross modal generation which generates images according to given texts. It extracts the common modal data from texts and transfers the semantic data into images. Text-to-image synthesis plays a more and more important role in computer vision. Images were edited by images in the past. With the development of text-to-image synthesis, images can also be edited by text, which greatly expands the application of computer vision. Text-to-image synthesis can be widely applied in human-computer interaction, such as cross modal retrieval [1] and artistic creation [2, 3].

Traditional text-to-image synthesis used variational autoencoder (VAE), attention mechanism, and recurrent neural network (RNN) to generate images step by step [4, 5]. Limited by generative ability of VAE, generated images are not as clear as real images. A new generative model GAN was proposed by Goodfellow et al. in 2014 [6]. GAN becomes a popular model in image generation task due to its strong

generating ability. Reed et al. [7] proved that GAN could be used to generate clear images from text description and proposed GAN-int-cls. It uses DCGAN as the backbone, text embedding, and random noises as inputs of the generator. The generated images, text embedding, and real images are inputs of the discriminator. Subsequently, many sophisticated models were proposed. These models can generate images according to general text, scene graph, or dialog. The quality of generated images has been improved a lot.

However, these models introduced many constraints and modules to generate realistic images. These will greatly increase parameters and floating-point operations per second (flops) of models. It will require more and more hardware resources (CPU, GPU, memory, and bandwidth) to deploy these models. High complexity also leads to high latency. This greatly limits application of text-to-image GAN in mobile terminal. It is necessary to compress text-to-image GAN. Canonical polyadic decomposition (CPD) is an easy and efficient way to compress and accelerate model in tensor decomposition. Many implementations of convolutional

neural networks (CNN) compression based on CPD [7–9] have already been proposed.

In this paper, we propose a general compressed architecture CPGAN for designing text-to-image GAN to reduce parameters and flops. CPGAN redesigns each layer of the original neural network by using CPD. The original convolution layer is decomposed into three convolution layers with different ranks and small size. A layer with a smaller rank has few parameters. According to the needs of the application, we can design architectures with different compression ratios by setting different ranks. During the training process of models with different ranks, it is time-consuming to select the appropriate learning rate. To this end, we use cyclical learning rate (CLR) [11] method to select the optimal learning rate for the redesigned architecture. In addition, GAN has the problem of unstable training. CPGAN is a deeper architecture than the classical GAN and is difficult to train from scratch. To solve this problem, we add conditioning augmentation module and introduce the idea of autoencoder method.

Our contributions can be summarized as follows:

- (i) We propose CPGAN to reduce parameters and maintain the generative ability of text-to-image GAN. It is a general method to design the light-weight architecture of GAN.
- (ii) To reduce high resource consumption caused by decomposition operation, we train CPGAN from scratch and do not need to pretrain the model. To the best of our knowledge, it is the first time to use CPD to design text-to-image GAN without using pretrained model.
- (iii) To stable the end-to-end training, we introduce the idea of autoencoder. The added decoder modules can be removed after training.

Experimental results on two representative cross modal datasets (Oxford-102 and CUB) prove that our architecture CPGAN can maintain the quality of generated images and reduce parameters and flops of original model effectively at the same time. In Oxford-102 and CUB, CPGAN performs better in inception score (IS) and Fréchet inception distance (FID) than original model. It reduces 8.8×10^9 flops and 1.31×10^6 parameters in Oxford-102. These show that our architecture can efficiently redesign text-to-image GAN without loss of image quality.

The rest of the paper is organized as follows. The work related to our paper is introduced in Section 2. In Section 3, we propose the efficient architecture CPGAN of text-to-image generative adversarial networks (GAN) based on canonical polyadic decomposition (CPD). Section 4 describes experimental settings and experimental results. Finally, we conclude this paper in Section 5.

2. Related Work

2.1. Canonical Polyadic Decomposition. The essence of neural network is the matrix transformation process of input data matrix using weight parameters. Each layer of neural

network is a large tensor, which can be decomposed into several small tensors. Canonical polyadic decomposition (CPD) is a standard tensor decomposition method. It was proposed by Hitchcock in 1927 [12]. It can decompose a tensor into a sum of rank-one tensors. CPD has been applied in psychometrics [13], signal processing [14], computer vision [15], data mining [16], and elsewhere. It also performs well in model compression.

Denton et al. [8] used CPD to approximate the original convolution kernel and presented two methods of improving approximation criterion. They performed fine-tuning on the decomposed kernels by fixing other layers. Jaderberg et al. [9] applied CPD to decompose a 4D kernel into two small kernels and use two methods to reconstruct the original filters. Lebedev et al. [10] used CPD to decompose the 4D convolution kernel tensor into four small kernels with nonlinear least squares and then replace original layer. Then, they fine-tuned the entire network using backpropagation. Lebedev et al.'s [10] method accelerated the second convolutional layer of AlexNet by 6.6 times at the cost of 1% accuracy loss. This exceeded the other two works, where Denton et al. [8] got 2 times speed-up and Jaderberg et al. [9] got 4.5 times speed-up at the cost of 1% accuracy loss.

Astrid et al. [17] proposed a CNN compression method based on CPD: CP-TPM. It achieved 6.98 times parameter reduction and 3.53 times speeding-up in AlexNet. It is better than the Tucker-based method [18] in the same network. Zhang et al. [19] and Tai et al. [20] also applied CPD to compress CNN. Original layers are pretrained to minimize the difference between the decomposed layer and the original tensor in the models of Astrid et al. [17], Zhang et al. [19], and Tai et al. [20]. Because CP decomposition operation consumes extensive resources, we do not decompose the pretrained weight tensor, but directly use CP decomposition to design an efficient architecture in text-to-image GAN.

2.2. Text-to-Image Synthesis. Text-to-image synthesis is a branch of computer vision which generates images according to given texts. It can be used for image editing, cross modal retrieval, and artistic creation. GAN has strong generating ability. It can generate realistic images and has been widely used for image generation. Since Reed et al. [7] first successfully used GAN for text image generation, GAN has also become a popular model in text-to-image synthesis.

Reed et al. [7] proposed GAN-int-cls by revising DCGAN and successfully generated plausible 64×64 images of birds and flowers from texts. In order to produce high resolution images, multiple stages generating was introduced into text-to-image synthesis, such as StackGAN [21], StackGAN++ [22], HDGAN [23], and LAPGAN [24]. StackGAN [21] stacked two conditional GANs to generate high resolution and plain images in two stages. Multiple generators were used to generate images of different scales using tree structure in StackGAN++ [22]. HDGAN [23] adopted hierarchically-nested discriminators to help the single-stream generator generate high resolution images.

LAPGAN [24] put forward a Laplacian pyramid framework through integrating a set of generators.

Xu et al. [25] and Qiao et al. [26] added attention mechanisms to synthesize image with fine-grained details. Besides, Reed et al. [27] adapted bounding box and key part information to improve quality of generated images. ACGAN [28] and TAC-GAN [29] used auxiliary class information to generate diversity images. Because these models show excellent cross modal generative ability, text-to-image GAN has been used for image editing [30, 31], cross modal retrieval [1], story visualization [2], and painting [3]. However, these models are too complicated to be deployed on the mobile end. To this end, we propose an end-to-end compression framework based on CPD. Compared to Shu et al. [32] and Li et al. [33], we do not need to pretrain GAN model. We design and train the compression model from scratch.

3. Canonical Polyadic Generative Adversarial Networks (CPGAN)

In this section, we introduce the designing of the efficient architecture (CPGAN) and the training process. Section 3.1 describes how to replace 4-dimensional convolutional weight tensors with three small kernels. Section 3.2 describes techniques for stabilizing training process of the redesigned architecture.

3.1. Canonical Polyadic Decomposition. GAN consists of a generator and a discriminator in general, both of which are convolutional neural networks. The weight tensor for convolution is a 4-dimensional tensor $\mathcal{W} \in \mathbb{R}^{K \times K \times S \times T}$, which maps input $\mathcal{X} \in \mathbb{R}^{I \times J \times S}$ into another representation $\mathcal{Y} \in \mathbb{R}^{X \times Y \times T}$. It can be written as

$$\mathcal{Y}(x, y, t) = \sum_{i=1}^K \sum_{j=1}^K \sum_{s=1}^S \mathcal{W}(k, k, s, t) \mathcal{X}(i, j, s), \quad (1)$$

where the first two dimensions of $\mathcal{W}(k, k, s, t)$ are the spatial dimension (K is typically 3 or 5), the third dimension is the input channel, and the fourth dimension is the output channel.

CPD is an approximation method which decomposes a tensor into a sum of rank-one tensors. In CPD, tensor $\mathcal{W} \in \mathbb{R}^{K \times K \times S \times T}$ can be represented as

$$\mathcal{W}(i, j, s, t) = \sum_{r=1}^R \mathcal{W}_{i,r}^{(1)} \mathcal{W}_{j,r}^{(2)} \mathcal{W}_{s,r}^{(3)} \mathcal{W}_{t,r}^{(4)}, \quad (2)$$

where R is the tensor rank and it is the sum of rank-one tensors, $\mathcal{W}_{i,r}^{(1)}$, $\mathcal{W}_{j,r}^{(2)}$, and $\mathcal{W}_{s,r}^{(3)}$, $\mathcal{W}_{t,r}^{(4)}$ are tensors of size $K \times R$, $K \times R$, $S \times R$, $T \times R$, respectively. Rank-one tensor is the vector outer product. Rank selection decides the compression ratio and it is a NP-hard problem in rank decomposition.

In convolutional layer, spatial dimension K does not have to be decomposed because the benefits of spatial decomposition are quite small. By using the variant of CP decomposition, tensor can be decomposed as

$$\mathcal{W}(i, j, s, t) = \sum_{r=1}^R \mathcal{W}_{s,r}^{(1)} \mathcal{W}_{i,j,r}^{(2)} \mathcal{W}_{t,r}^{(3)}, \quad (3)$$

where $\mathcal{W}_{i,j,r}^{(2)}$ is a tensor of size $K \times K \times R$. Substituting equation (3) into equation (2), we obtain the following approximate representation of the convolution:

$$\mathcal{Y}(x, y, t) = \sum_{r=1}^R \mathcal{W}_{t,r}^{(3)} \left(\sum_{j=1}^D \sum_{i=1}^D \mathcal{W}_{r,j,i}^{(2)} \left(\sum_{s=1}^R \mathcal{W}_{r,s}^{(1)} \mathcal{X}(i, j, s) \right) \right). \quad (4)$$

Performing rearranging and combining, we can get the following three consecutive expressions:

$$\mathcal{Y}^{(1)}(i, j, r) = \sum_{s=1}^S \mathcal{W}_{r,s}^{(1)} \mathcal{X}(i, j, s), \quad (5)$$

$$\mathcal{Y}^{(2)}(x, y, r) = \sum_{j=1}^D \sum_{i=1}^D \mathcal{W}_{r,j,i}^{(2)} \mathcal{Y}^{(1)}(i, j, r), \quad (6)$$

$$\mathcal{Y}(x, y, t) = \prod_{r=1}^R \mathcal{W}_{t,r}^{(3)} \mathcal{Y}^{(2)}(x, y, r), \quad (7)$$

where $\mathcal{Y}^{(1)}$ and $\mathcal{Y}^{(2)}$ are the intermediate tensors of sizes $I \times J \times R$ and $I' \times J' \times R$, respectively. The original big layer can be decomposed into three small layers, as shown in Figure 1. For example, the third convolution layer of GAN-int-cls has 128 input channels, 512 output channels, and 3×3 filters ($128 \times 512 \times 3 \times 3$); we can decompose it into three convolution layers with the following parameters: $128 \times R \times 1 \times 1$, $R \times R \times 3 \times 3$, and $R \times 512 \times 1 \times 1$. R is the rank which can be set as different values according to the need of tasks.

3.2. Overall Framework. We take the classical model GAN-int-cls as the original model to compress. This model has the most compact structure and parameters. The main convolution layers of the generator in other text-to-image GAN models are similar to GAN-int-cls. We redesign GAN-int-cls to show the effectiveness and generality of our compression architecture. As shown in Figure 2, the proposed CPGAN contains two novel components which can stabilize the training of decomposed GAN: conditioning augmentation and autoencoder module.

Conditioning augmentation (CA) is proposed by Zhang et al. [21] which alleviates the difficulty of GAN training caused by text embedding sparsity. CA is to randomly sample the hidden variables as the input of the generator from the independent Gaussian distribution $\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))$. φ_t is the text embedding which is generated by encoding text description. $\mu(\varphi_t)$ and $\Sigma(\varphi_t)$ are the mean and diagonal covariance matrix functions of the text embedding φ_t , respectively. We use pretrained char-CNN-RNN [34] to get the text embedding φ_t . Then, we feed φ_t into CA and obtain $\mu(\varphi_t)$ and $\Sigma(\varphi_t)$. Similar to StackGAN [21], we also add the Kullback-Leibler (KL) divergence into our training objectives, which is the KL

FIGURE 1: Original convolution layer and decomposed three convolution layers based on CPD. The top figure is the original filter of size $D \times D \times S \times T$. The bottom figure is the three decomposed filters of sizes of $1 \times 1 \times S \times R$, $K \times K \times R \times R$, and $1 \times 1 \times R \times T$. The three decomposed filters can approximate the original filter.

FIGURE 2: The architecture of CPGAN. The three encoding layers in the blue dotted box are the three decomposed filters obtained by decomposing the original convolution layer.

divergence between the standard Gaussian distribution $\mathcal{N}(0, I)$ and the conditioning Gaussian distribution $\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))$, as shown in the following equation:

$$D_{\text{KL}}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t)) \| \mathcal{N}(0, I)). \quad (8)$$

Autoencoder (AE) is used for representation learning by reconstructing input. The decomposed architecture is deeper than the original model, which increases the instability of training. So, we use AE to stabilize the training process. AE is composed of an encoder and a decoder in general. We regard each convolution layer as an encoder and add a decoder corresponding to each convolution layer. The training objective of AE is the reconstruction loss. We use mean square error (MSE) $\|x_1 - h(x_1)\|_2^2$ as the AE loss, where x_1 is the input of layer and $h(\cdot)$ is the function of AE. The decoder will be removed after training.

The generator objective of original GAN-int-clis contains matching-aware loss and interpolation loss, as shown in

$$G_{\text{ori}} = G_1(z, t) + G_2(z, \beta t_1 + (1 - \beta)t_2), \quad (9)$$

where z is the random noise, t_1 and t_2 are text embeddings, and β is a decimal between 0 and 1 and used to interpolate between text embeddings t_1 and t_2 .

In the generator objective of our model, we add KL divergence and MSE reconstruction loss into the original model objective, as shown in the following equation:

$$G = G_{\text{ori}} + D_{\text{KL}}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t)) \| \mathcal{N}(0, I)) + \|x_1 - h(x_1)\|_2^2. \quad (10)$$

The discriminator objective of the original model and our model is both matching-aware loss:

$$D = D_1(x, t) + D_2(x, \hat{t}) + D_3(G(z), t). \quad (11)$$

We use the above scheme to train an efficient architecture from scratch. The training algorithm is shown in Algorithm 1. Firstly, original convolutions are decomposed into three layers through equations (5)–(7). Secondly, each layer is regarded as an encoder and a decoder is added corresponding to each layer. Thirdly, we encode matching text t and mismatching text $\hat{t}$ and get text embeddings. Then, we use CA to process text embeddings and get independent Gaussian distribution. From the independent Gaussian distribution, we sample variables and concatenate it with random noise. The following training process is the same as GAN-int-cls with different training objectives of generator. The objective function of our model adds the loss of CA and autoencoder on the basis of the original model's objective function. Until the training is finished, we remove added decoder layers and obtain the model of CPGAN.

4. Experiments

We conduct extensive experiments to evaluate the proposed CPGAN. In Section 4.1, we introduce the experimental dataset and evaluation index. Section 4.2 describes the setting of learning rate and the other experimental hyperparameters. In Section 4.3, we compare our CPGAN with previous GAN-int-cls models for text-to-image synthesis.

4.1. Overall Framework. To show the generality of our method, we choose the classic model GAN-int-cls as our original model. Same as GAN-int-cls, our method is evaluated on CUB [35] and Oxford-102 [36]. The CUB dataset covers 200 kinds of birds, including 5,994 training images and 5,794 test images. In addition to category labels, each image contains bounding box, bird key part of bird information, and bird attributes. Oxford-102 flowers dataset is a flower dataset which contains 8,189 images. It is divided into 102 categories and each category contains 40 to 258 images. Each image has large scale, pose and light variations. The dataset is divided into a training set, a validation set, and a test set. Both datasets are benchmark image datasets and each image corresponds to 10 single sentence descriptions.

In order to evaluate our model, we use inception score (IS) and Fréchet inception distance (FID) to evaluate the quality of the generated images. IS uses pretrained InceptionNet-V3 to judge whether the generated image is clear and diverse. High IS score means that images are clear and diverse. FID calculates feature distance between the real image and the fake image as a supplement of IS evaluation index. These two indicators are widely used to evaluate the quality of generated images.

4.2. Implementation Details. Learning rate is a very important hyperparameter in deep learning. Reasonable learning rate can make the model converge to the minimum point instead of the local optimal point or saddle point. In this paper, we use the method CLR [11] and MultistepLR to set learning rate and learning rate attenuation.

CLR was proposed by Smith. It changes learning rate periodically in the iterative process, rather than a fixed value. It is used to find the optimal learning rate automatically instead of manual experiments. We use CLR to get a learning rate setting. CLR method needs to set three parameters, minimum learning rate (min_lr), maximum learning rate (max_lr), and iteration. min_lr and max_lr are the smallest value and the biggest value of learning rate, respectively. Iteration is the number of test iterations at each learning rate. We increase the learning rate from 0.00001 to 0.001 and get the loss curve under different learning rates (see Figure 3).

We choose the appropriate learning rate according to maximum absolute slope criterion. According to Figure 3, we select 0.0002 and 0.00015 as the learning rates of the Oxford-102 dataset and 0.0001 and 0.00008 as the learning rates of the CUB dataset.

MultistepLR is a learning rate attenuation method in PyTorch. It has three hyperparameters: initial learning rate (ini_lr), epoch to update learning rate (epo), and multiplication factor(mfc). ini_lr is the initial learning rate during the training. epo is the epoch when we change the learning rate. mfc is the attenuation coefficient of learning rate. In the experiment using MultistepLR, the initial learning rate is *ini_lr*. When the experiment runs *epo* epochs, the learning rate is changed to *ini_lr* * mfc.

In this paper, we set the MultistepLR hyperparameters ini_lr, epo, and mfc as 0.0001, 600, and 0.8 in CUB and 0.0002, 600, and 0.75 in Oxford-102. The batch size in our experiment is 64. The optimizer of CPGAN is Adam [37] with momentum of 0.5.

4.3. Comparison with Original Model. In CP decomposition, rank represents compression ratio and it is hard to select. Due to the need of text-to-image synthesis task, we design the lightweight model on the premise of ensuring the quality of generated images. We do extensive experiments to balance the performance and the compression ratio.

As shown in Table 1, we do a large number of experiments to find the balance. The ratio is rank ratio, where 1.0 is full rank decomposition and 0.9 means about 0.9 times of original layer's number of input channels. A layer with a smaller rank has few parameters. Table 1 shows that with increasing of rank, flops and parameters grow. When the decomposition rank is close to 0.7, the parameters begin to exceed the original model's parameters (5.76×10^6). With the increase of rank, the quality of images generated by the model has not been greatly improved. The value of FID decreases first and then changes slightly with the increase of model parameters, while IS is not stable. It may be that the calculation of IS needs to use the edge distribution of data, but generated samples in Oxford-102 are not enough to get accurate edge distribution.

As shown in Table 1, FID gets the best value when rank ratio is 0.5. The model is compressed by about 23% parameters and 29% flops. The generated images are better than the original model on FID and IS. It can prove that our method can generate better images with less parameters than the original

Input: mini-batch images x , text description t , and number of training batch steps S .

Output: CPGAN model.

- (1) Use equations (5)–(7) to decompose the original convolutional layer in generator;
- (2) Add CA module for text embedding and add decoders layers;
- (3) Select an appropriate learning rate for the decomposed model;
- (4) **For** $N = 1$ **to** S **do**
- (5) Encode text description into embedding t ;
- (6) Feed t into CA and obtain $\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))$;
- (7) Sample $\hat{c}$ from $\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))$ and random noise z ;
- (8) Concatenate z and $\hat{c}$ and feed it into the generator;
- (9) Update discriminator D by equation (11);
- (10) Update generator G by equation (10);
- (11) **End for**
- (12) Discard all decoders and get a trained CPGAN.

ALGORITHM 1: Overall scheme of CPGAN algorithm.

FIGURE 3: Selection for appropriate learning rate for CPGAN. (a) Loss of different learning rates on Oxford-102. (b) Loss of different learning rates on CUB.

TABLE 1: Experimental results of different rank ratios for the CPGAN in Oxford-102.

Ratio	FID	IS	Flops	#Parameters
0.1	158.74	3.26 ± 0.05	1.20×10^{10}	2.91×10^6
0.2	98.06	3.43 ± 0.06	1.35×10^{10}	3.13×10^6
0.3	85.08	2.98 ± 0.05	1.54×10^{10}	3.46×10^6
0.4	81.17	2.96 ± 0.04	1.80×10^{10}	3.90×10^6
0.5	74.69	3.04 ± 0.05	2.14×10^{10}	4.45×10^6
0.6	77.59	3.54 ± 0.06	2.53×10^{10}	5.10×10^6
0.7	76.50	2.80 ± 0.06	3.00×10^{10}	5.87×10^6
0.8	79.04	3.56 ± 0.05	3.53×10^{10}	6.74×10^6
0.9	76.97	3.17 ± 0.05	4.14×10^{10}	7.72×10^6
1.0	77.07	3.23 ± 0.05	4.83×10^{10}	8.82×10^6

model. It is effective to use CP decomposition to reconstruct the model and design compact text-to-image GAN without loss of image quality. Although 8.8×10^9 flops and 1.31×10^6 parameters are reduced, the images generated by CPGAN get a

little improvement on IS and FID. This shows that the image generated by the model with more parameters may not be better. So around the rank of 0.5, we look for a better model ensuring the quality of generated images.

Table 2 shows the comparison between our best generative model and the original model on IS, FID, parameters, and flops. FID and IS of the original model are 79.55 and 2.66 ± 0.03 in Oxford-102, while those of our best model are 74.40 and 3.68 ± 0.08 , respectively. In CUB, the images generated by our best model get 65.94 on FID and 5.03 ± 0.07 on IS, while those of original model are 68.79 and 2.88 ± 0.04 , respectively. The comparison of representative images on Oxford-102 and CUB dataset can be seen in Figures 4 and 5, respectively. The better generated images of CPGAN indicate that our proposed method can generate more realistic images from text descriptions. These results also prove that there are redundant parameters in existing text-to-image GAN. A more concise and efficient text-to-image GAN model can be designed based on CPD.

TABLE 2: Comparison between our model and the original model.

Model	Oxford-102		CUB		Flops	#Parameters
	FID	IS	FID	IS		
Original	79.55	2.66 ± 0.03	68.79	2.88 ± 0.04	3.02×10^{10}	5.76×10^6
Redesigned	74.40	3.68 ± 0.08	65.94	5.03 ± 0.07	2.33×10^{10}	5.07×10^6

FIGURE 4: Generated images by our proposed model and the original model on Oxford-102.

FIGURE 5: Generated images by our proposed model and the original model on CUB.

5. Conclusions

In this paper, we propose a simple and efficient architecture CPGAN based on CPD. CPGAN can reduce extensive parameters and flops of the original model. It also improves the quality of generated images at the same time. In the process of designing CPGAN model, we replace the convolution layer with three CP decomposed small layers to achieve a certain compression. In order to stabilize the training process, we introduce conditioning augmentation to reduce the instability caused by text embedding sparsity. To further improve the end-to-end training of our model, the idea of autoencoder is integrated into the model. Each decomposed layer can be regarded as an encoder layer and is paired with an added decoder layer. The decoder layers can be removed after training. Experiments demonstrate that CPGAN reduces about 23% parameters and 29% flops with a little improvement of generated image quality in Oxford-102.

Extensive experimental results demonstrate that our proposed CPGAN can design an efficient text-to-image GAN. We have also decomposed similar convolution layers in other GAN models and these experiment results were similar to the experiment results of GAN-int-cls. The main convolution layers of the generator in other text-to-image GAN models are similar to GAN-int-cls. It is applicable for other cross modal GANs to use CPD. In the existing methods, the rank is set manually, which is time-consuming. Therefore, the automatic selection of rank may be a research direction in the future.

Data Availability

The datasets used in this paper are public datasets which can be accessed via the following websites: <http://www.vision.caltech.edu/visipedia/CUB-200-2011.html> and <https://www.robots.ox.ac.uk/~vgg/data/flowers/102/>

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

- [1] J. Gu, J. Cai, S. R. Joty et al., "Look, imagine and match: improving textual-visual cross-modal retrieval with generative models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7181–7189, Salt Lake City, UT, USA, 2018.
- [2] Y. Li, Z. Gan, Y. Shen et al., "StoryGAN: a sequential conditional GAN for story visualization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6329–6338, Seattle, WA, USA, 2019.
- [3] W. Li, P. Zhang, L. Zhang et al., "Object-driven text-to-image synthesis via adversarial training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12174–12182, Seattle, WA, USA, 2019.
- [4] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Generating images from captions with attention," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, 2016.
- [5] A. van den Oord, N. Kalchbrenner, L. Espeholt et al., "Conditional image generation with PixelCNN decoders," *Advances in Neural Information Processing Systems*, vol. 29, pp. 4790–4798, 2016.
- [6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 2, pp. 2672–2680, 2014.
- [7] S. Reed, Z. Akata, X. Yan et al., "Generative adversarial text to image synthesis," *Proceedings of Machine Learning Research*, vol. 48, pp. 1060–1069, 2016.
- [8] E. Denton, W. Zaremba, J. Bruna et al., "Exploiting linear structure within convolutional networks for efficient evaluation," *Advances in Neural Information Processing Systems*, vol. 27, pp. 1269–1277, 2014.
- [9] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *Proceedings of the British Machine Vision Conference*, 2014.
- [10] V. Lebedev, Y. Ganev, M. Rakhuba et al., "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," 2014, <http://arxiv.org/abs/1412.6553>.
- [11] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proceedings of the 2017 IEEE Winter Conference On Applications Of Computer Vision (WACV)*, pp. 464–472, Santa Rosa, CA, USA, 2017.
- [12] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.
- [13] P. M. Kroonenberg, *Applied Multiway Data Analysis*, Wiley-Interscience, Hoboken, NJ, USA, 2008.
- [14] F. Roemer and M. Haardt, "A semi-algebraic framework for approximate CP decompositions via simultaneous matrix diagonalizations (SECSI)," *Signal Processing*, vol. 93, no. 9, pp. 2722–2738, 2013.
- [15] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: tensorfaces," in *Proceedings of the European Conference on Computer Vision*, pp. 447–460, Copenhagen, Denmark, 2002.
- [16] E. Acar, S. A. Çamtepe, M. S. Krishnamoorthy et al., "Modeling and multiway analysis of chatroom tensors," in *Proceedings of the International Conference On Intelligence And Security Informatics*, pp. 256–268, San Diego, CA, USA, 2005.
- [17] M. Astrid and S. I. Lee, "CP-decomposition with tensor power method for convolutional neural networks compression," in *Proceedings of the 2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 115–118, Jeju, South Korea, 2017.
- [18] Y. D. Kim, E. Park, S. Yoo et al., "Compression of deep convolutional neural networks for fast and low power mobile applications," *Computer Science*, vol. 71, no. 2, pp. 576–584, 2015.
- [19] Q. Zhang, L. T. Yang, Z. Chen et al., "An improved deep computation model based on canonical polyadic decomposition," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 10, pp. 1657–1666, 2017.
- [20] C. Tai, T. Xiao, Y. Zhang et al., "Convolutional neural networks with low-rank regularization," 2015, <http://arxiv.org/abs/1404.3978>.
- [21] H. Zhang, T. Xu, H. Li et al., "StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5907–5915, Venice, Italy, 2017.
- [22] H. Zhang, T. Xu, H. Li et al., "StackGAN++: realistic image synthesis with stacked generative adversarial networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1947–1962, 2018.
- [23] Z. Zhang, Y. Xie, and L. Yang, "Photographic text-to-image synthesis with a hierarchically-nested adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6199–6208, Salt Lake City, UT, USA, 2018.
- [24] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *Advances in Neural Information Processing Systems*, vol. 28, pp. 1486–1494, 2015.
- [25] T. Xu, P. Zhang, Q. Huang et al., "AttnGAN: fine-grained text to image generation with attentional generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1316–1324, Salt Lake City, UT, USA, 2018.
- [26] T. Qiao, J. Zhang, D. Xu et al., "Learn, imagine and create: text-to-image generation from prior knowledge," *Advances in Neural Information Processing Systems*, pp. 887–897, 2019.
- [27] S. E. Reed, Z. Akata, S. Mohan et al., "Learning what and where to draw," *Advances In Neural Information Processing Systems*, pp. 217–225, 2016.
- [28] A. Odena and C. Olah, "Conditional image synthesis with auxiliary classifier GANs," in *Proceedings of the International Conference on Machine Learning*, pp. 2642–2651, Sydney, Australia, 2017.
- [29] A. Dash, J. C. Gamboa, S. Ahmed et al., "TAC-GAN - text conditioned auxiliary classifier generative adversarial network," 2017, <http://arxiv.org/abs/1703.06412>.
- [30] H. Dong, S. Yu, C. Wu et al., "Semantic image synthesis via adversarial learning," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5706–5714, Venice, Italy, 2017.
- [31] D. M. Vo and A. Sugimoto, "Paired-D GAN for semantic image synthesis," in *Proceedings of the Asian Conference on Computer Vision*, pp. 468–484, Salt Lake City, UT, USA, 2018.
- [32] H. Shu, Y. Wang, X. Jia et al., "Co-evolutionary compression for unpaired image Translation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3234–3243, Seoul, Korea, 2019.

- [33] M. Li, J. Lin, Y. Ding et al., “GAN compression: efficient architectures for interactive conditional GANs,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5283–5293, Seoul, Korea, 2020.
- [34] S. Reed, Z. Akata, H. Lee et al., “Learning deep representations of fine-grained visual descriptions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 49–58, Las Vegas, NV, USA, 2016.
- [35] C. Wah, S. Branson, P. Welinder et al., “The Caltech-UCSD birds-200-2011 dataset,” Computation & Neural Systems Technical Report, 2011.
- [36] M. E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” in *Proceedings of the 2008 Sixth Indian Conference On Computer Vision, Graphics & Image Processing*, pp. 722–729, IEEE, Bhubaneswar, India, 2008.
- [37] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” in *Proceedings of the International Conference On Learning Representations*, San Diego, CA, USA, 2015.