IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 30, 2020, accepted September 12, 2020, date of publication September 30, 2020, date of current version October 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027800

Generating Images in Compressed Domain Using Generative Adversarial Networks

BYEONGKEUN KANG[1,2], SUBARNA TRIPATHI[3], AND TRUONG Q. NGUYEN[4], (Fellow, IEEE)

[1]Department of Electronic and IT Media Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea
[2]Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, South Korea
[3]Intel Labs, San Diego, CA 92131, USA
[4]Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093, USA

Corresponding author: Byeongkeun Kang (byeongkeun.kang@seoultech.ac.kr)

This work was supported in part by the NSF under Grant IIS-1522125, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1A6A1A03032119.

ABSTRACT In this article, we present a generative adversarial network framework that generates compressed images instead of synthesizing raw RGB images and compressing them separately. In the real world, most images and videos are stored and transferred in a compressed format to save storage capacity and data transfer bandwidth. However, since typical generative adversarial networks generate raw RGB images, those generated images need to be compressed by a post-processing stage to reduce the data size. Among image compression methods, JPEG has been one of the most commonly used lossy compression methods for still images. Hence, we propose a novel framework that generates JPEG compressed images using generative adversarial networks. The novel generator consists of the proposed locally connected layers, chroma subsampling layers, quantization layers, residual blocks, and convolution layers. The locally connected layer is proposed to enable block-based operations. We also discuss training strategies for the proposed architecture including the loss function and the decoding between its generator and its discriminator. The proposed method is evaluated using the publicly available CIFAR-10 dataset and LSUN bedroom dataset. The results demonstrate that the proposed method is able to generate compressed data with competitive qualities.

INDEX TERMS Generative adversarial networks, image generation, image synthesis.

I. INTRODUCTION

Most images and videos exist in a compressed form since data compression saves lots of data storage and network bandwidth and further enables many applications such as real-time video streaming in cell phones or virtual reality (VR) devices. Compression is indeed crucial, considering that compressed image and video can be about 10 times and 50 times smaller than raw data, respectively. Nevertheless, typical generative adversarial networks (GAN) focus on generating raw RGB images or videos [1]–[6]. Considering one of the most common usages of GANs is generating large-scale synthetic images/videos for data augmentation, the created images/videos often require compression in a post-processing stage to store the large dataset in hardware [7]–[10]. Besides, typical GANs are evaluated using the generated raw RGB data although compression is processed to the raw data prior to final applications. Hence, we investigate GAN frameworks that aim to generate compressed data and to evaluate the networks using the generated encoded images.

One application of the frameworks that generate compressed data is creating/modifying content for VR/AR devices in real-time. Let's consider the circumstance that a user wants to put a magenta cowboy hat on a person in VR/AR content. Then, a traditional service provider should have the specific type of hat in their database so that they can provide the hat upon a user's request in real-time. Also, the hat usually needs additional processing to be in a proper orientation and size with the consideration of the user's content. Given properly trained GANs, the service provider does not need the hat in their database and does not demand the processing for rotation since the specific type/orientation of hat can be generated by providing a corresponding noise in the latent space and by forward-propagating the noise through the networks.

The associate editor coordinating the review of this manuscript and approving it for publication was Omar Sultan Al-Kadi.

As a trained model can generate almost infinitely diverse images, a model can substitute a huge database. Furthermore, while a typical GAN framework requires compression of generated images to reduce transmitting content's size to VR/AR device, the framework that generates compressed data does not need the compression step and can immediately transmit the generated content to VR/AR device. Thus, the framework can be utilized for the real-time creation/modification of VR/AR content without requiring collecting/storing huge databases and processing compression to transmit.

We focus on the GAN frameworks for compressed image generation since image generation networks [1]–[4] have been far more studied comparing to video generation networks [5], [6]. In image compression, JPEG [11], [12] has been one of the most commonly used lossy compression methods [13]–[15]. It has been used in digital cameras and utilized for storing and transmitting images. While JPEG has many variants, typical JPEG consists of color space transformation, chroma subsampling, block-based discrete cosine transform (DCT) [16], quantization, and entropy coding [17]. The compression method first converts an image in the RGB domain to the image in another color space (YCbCr) that separates luminance and chrominance components. Then, the chrominance components are downsampled. It then applies the 8 × 8 block-based DCT to both the luminance component and the subsampled chrominance components to represent them in the frequency domain. It discards details of high-frequency information by applying quantization. Lastly, the processed data is stored using entropy coding.

We argue that investigating the frameworks of generating compressed images is important to accomplish the creation of more visually plausible large-scale images that require storing in a compressed domain. Typical GAN frameworks optimize and select the networks' architectures and parameters (weights) based on generated raw RGB images. Accordingly, if we take into account the compression process in the post-processing stage, the choice might not be the optimal decision. Hence, we propose to optimize/determine architectures and parameters by evaluating them using generated images in a compressed domain.

We propose a novel framework that generates compressed images using generative adversarial networks as shown in Figure 1. The framework consists of a generator, a discriminator, and a decoder between the generator and the discriminator. The proposed generator produces compressed data given a randomly selected noise in a latent space. The decoder is applied to make the data from the generator and the training data to be in the same domain. Since the generator outputs compressed data and the training data is raw RGB images, the decoder converts encoded data to a raw image. The discriminator takes synthesized images and real images and aims to differentiate them.

The proposed generator has three paths, one path for the luminance component and the other two paths for the chrominance components. The separate paths are proposed to process any required chroma subsampling. We also propose the locally connected layer that takes an input of a subregion and produces the output for the corresponding subregion. The proposed locally connected layer is able to handle block-based processing in JPEG compression. In summary, the contributions of our work are as follows:

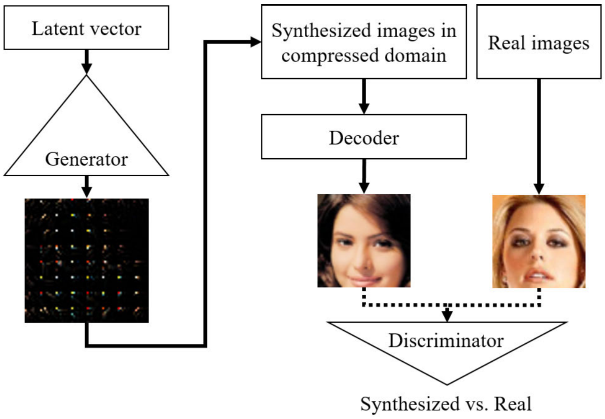


FIGURE 1. The proposed framework to generate compressed images. The framework consists of a generator, a discriminator, and a decoder between the generator and the discriminator. The visualized generator output is an intermediate example.

- We propose the framework that generates JPEG compressed images using generative adversarial networks.
- We propose the locally connected layer to enable block-based operations in JPEG compression.
- We analyze the effects of compression for the proposed method and other methods.

II. RELATED WORKS

A. GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks were introduced in [1] where the framework is to estimate generative models by learning two competing networks (generator and discriminator). The generator aims to capture the data distribution by learning the mapping from a known noise space to the data space. The discriminator differentiates between the samples from the training data and those from the generator. These two networks compete since the goal of the generator is making the distribution of generated samples equivalent to that of the training data while the discriminator's objective is discovering the discrepancy between the two distributions.

While the work in [1] employed multilayer perceptrons for both generator and discriminator, deep convolutional GANs (DCGANs) replaced multilayer perceptrons by convolutional neural networks (CNNs) to take the advantage of shared weights, especially for image-related tasks [2]. To utilize CNNs in the GAN framework, extensive architectures are explored for relatively stable training. They examined stability even for models with deeper layers and for networks that generate high-resolution outputs. The analysis includes fractional-stride convolutions, batch normalization, and activation functions. Salimans *et al.* presented the methods that improve the training of GANs [18]. The techniques include matching expected activations of training data and those of generated samples, penalizing similar samples in a mini-batch, punishing sudden changes of weights, one-sided label smoothing, and virtual batch normalization.

Arjovsky *et al.* presented the advantage of the Earth-Mover (EM) distance (Wasserstein-1) comparing to other popular probability distances and divergences such as the Jensen-Shannon divergence, the Kullback-Leibler divergence, and the Total Variation distance [3]. The advantage of the EM distance is that it is continuous everywhere and differentiable almost everywhere when it is applied to a neural network-based generator with a constrained input noise variable. They also showed that the EM distance is a more sensible cost function. Based on these, they proposed Wasserstein GAN that uses a reasonable and efficient approximation of the EM distance. They then showed that the proposed GAN achieves improved stability in training. However, clipping weights for Lipschitz constraint in [3] might cause optimization difficulties [4]. Hence, Gulrajani *et al.* proposed penalizing the gradient norm to enforce Lipschitz constraint instead of clipping [4].

Wasserstein GAN trains its discriminator multiple times at each training of its generator so that the framework can train the generator using the more converged discriminator [3]. To avoid the expensive multiple updates of the discriminator, Heusel *et al.* proposed to use the two time-scale update rule (TTUR) [19] in a Wasserstein GAN framework [20]. Since TTUR enables separate learning rates for the discriminator and the generator, they can train the discriminator faster than the generator by selecting a higher learning rate for the discriminator comparing to that of the generator. It is also proved that TTUR converges to a stationary local Nash equilibrium under mild assumptions. They further experimentally showed that their method outperforms most other state-of-the-art methods. Hence, the proposed framework of this article is based on [20].

B. IMAGE COMPRESSION: JPEG

JPEG [12] has been one of the most commonly used lossy compression methods for still images with continuous tones [13]–[15]. It has been used for digital cameras, photographic images on the World Wide Web, medical images, and many other applications.

In JPEG, the discrete cosine transform (DCT) is utilized since it achieves high energy compaction while having low computational complexity. Considering an image contains uncorrelated (various) information, block-based DCT is used so that each block contains correlated data. Using a small block prevents from compressing correlated information. A large block with uncorrelated pixels increases computational complexity without compression gain. 8×8 block size is selected based on a psychovisual evaluation.

DCT transforms an 8×8 block of an image to 64 amplitudes of 2D cosine functions with various frequencies. Since the sensitivity of a human eye is different for each frequency, quantization is applied differently for each amplitude. The amplitudes for low-frequencies are maintained with high accuracy and those of high-frequencies are quantized using larger quantization values. Quantization is responsible for most of the information loss in JPEG.

After quantization, since most of the non-zero components are for low-frequencies, the amplitudes are encoded in zig-zag order using a value pair. The information is then encoded using Huffman coding considering the statistical distribution of the information [17].

While these are the baseline of JPEG, other additional methods and components were also suggested for particular purposes. Also, typical JPEG uses the YCbCr color space and chroma subsampling [11].

C. OTHER RELATED WORKS

Our work differs from learning neural networks for image compression such as autoencoders [21]–[24] that aims to learn image encoders to compress real images. The proposed method and learning encoders differ in two aspects. First, the goal of the proposed method is generating synthetic JPEG compressed data while that of the latter is compressing real images. Second, the proposed method intends to utilize existing standard decoder in general electronic devices while the latter requires a particular decoder to decode the compressed data.

This article is also distinct from [25], [26] that are about utilizing GANs for postprocessing real data in a frequency domain. The proposed work generates compressed images from random noises in the latent space.

D. OTHER APPLICATIONS OF GANS

While one of initial applications of GANs was in synthesizing images from noises, many other applications have also been introduced such as image/video super-resolution [27]–[30], image-to-image translation [31]–[34], inpainting [35]–[37], denoising [38], [39], text-to-image translation [40]–[42], artifact removal [43], [44], etc.

For image super-resolution, Ledig *et al.* introduced SR-GAN (Super-Resolution GAN) to recover high-frequency details [27]. Wang *et al.* then investigated network architecture, adversarial loss, and perceptual loss in SR-GAN and proposed an enhanced SR-GAN (ESR-GAN) [28]. Considering that low-/high-resolution image pairs are unavailable in general, Yuan *et al.* proposed an unsupervised learning-based method which also uses GAN as a basic component [29]. Recently, Lucas *et al.* introduced a GAN that is optimized for video super-resolution [30].

Isola *et al.* introduced a conditional adversarial network framework to solve image-to-image translation problems [31]. Zhu *et al.* then presented an approach to train adversarial networks without paired training data [34]. By assuming a shared-latent space, Liu *et al.* proposed an unsupervised framework based on Coupled GANs [33], [45]. To learn image-to-image translation between more than two domains, Choi *et al.* proposed StarGAN that can translate an

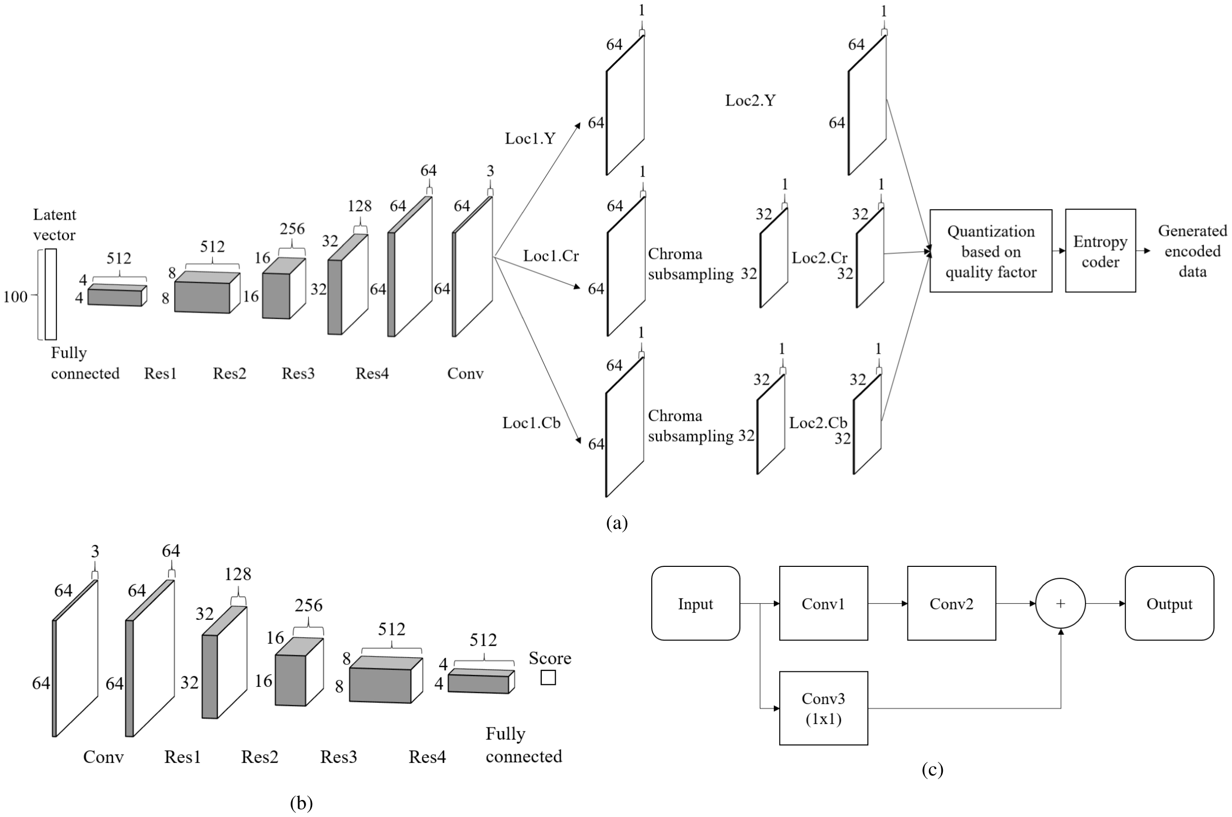


FIGURE 2. The proposed architecture for JPEG compressed image generation. (a) Generator. (b) Discriminator. (c) Residual block. The proposed generator consists of three paths, one for each luminance or chrominance component. The generator employs the proposed locally connected layer to operate block-based processing. The visualized generator considers chroma subsampling ratio of 4:2:0.

image to another image in one of the multiple domains using only a single model [32].

Considering image completion, Iizuka *et al.* proposed a GAN-based approach that consists of a completion network, a global discriminator, and a local discriminator [46]. The global discriminator and the local discriminator are to ensure global coherency and local consistency, respectively. Yu *et al.* then introduced a generative model-based method for inpainting multiple holes at arbitrary locations and with variable sizes in images [35]. For semantic image inpainting, Yeh *et al.* proposed a GAN-based method that processes using high-level context information of given images [36].

Reed *et al.* proposed a GAN-based framework for text-to-image synthesis [40]. Zhang *et al.* then introduced Stacked GANs to generate higher-resolution(256×256) images from text descriptions [41], [42].

Regarding denoising and artifact removal, Tripathi *et al.* proposed a GAN-based method for denoising corrupted face images [38]. Yang *et al.* also presented a GAN-based denoising framework for computed tomography (CT) images [39]. Galteri *et al.* proposed a GAN framework to eliminate compression artifact [43], [44].

III. PROPOSED METHOD

We propose a framework that generates JPEG compressed images using generative adversarial networks. We use the architectures analyzed in the TTUR method [20] as our baseline networks since the method achieves one of the state-of-the-art results. The generator in the baseline architecture consists of one fully connected layer, four residual blocks, and one convolution layer. The discriminator in the architecture consists of one convolution layer, four residual blocks, and one fully connected layer as shown in Figure 2(b). The residual block consists of two paths (see Figure 2(c)). One path has two convolution layers with filter-size of 3×3, and the other has only one convolution layer with filter-size of 1×1. All the convolution layers outside of residual blocks have filter-size of 3×3.

Given the baseline architecture, we propose an architecture and training strategy to generate JPEG compressed images in the framework of generative adversarial networks. We first propose a novel generator in Section III-A. The proposed generator has three paths, one for each luminance or chrominance component. The proposed generator also has additional layers including the proposed locally connected layers, chroma subsampling layer, quantization layer, and entropy encoding layer. The entropy encoding layer in the generator and the entropy decoding in Section III-B are excluded during training since the entropy encoding/decoding is lossless.

We then present the processing between the generator and the discriminator in Section III-B. Since typical generators generate RGB images that are in the same domain with the training images, any additional processing is not required to use the output of the generator for the input to the discriminator. However, since the proposed generator produces JPEG compressed data, the outputs of the generator and the training images are in different domains and cannot be used together for the discriminator. Consequently, we need to either compress the training images or decode the generated JPEG compressed data so that they are in the same domain.

In Section III-C, we discuss training strategies for the proposed architecture. Although many studies have been conducted to improve training stability, training GANs for non-typical images is still quite challenging.

A. GENERATOR

We propose a novel generator that generates JPEG compressed images (see Figure 2). The generator consists of six locally connected layers, two chroma subsampling layers, a quantization layer, and an entropy encoding layer in addition to the layers in the baseline generator. The generator has three paths where each path generates one of luminance or chrominance components in the YCbCr representation. The separated paths are required to handle any required chroma subsampling since the resolution of a luminance component and chrominance components are different if chroma subsampling is applied. The locally connected layer is proposed to operate block-based processing. The entropy encoding layer is not applied during training along with the corresponding decoding in the decoder since the encoding is lossless, so it does not impact the results. The quantization layer and the entropy encoding layer are not learned and follow the JPEG standard so that generated data can be decoded by using a standard JPEG decoder.

The proposed locally connected layer takes an input of a subregion and produces an output for the corresponding subregion (see Figure 3). The layer is proposed to perform operations comparable to block-based processing. Comparing to a convolution layer, the proposed layer is different since a convolution layer takes input from a region and outputs to only a single location. The nearby outputs from a convolution layer are produced by using different regions of inputs. In other words, to generate 8 × 8 outputs using a convolution layer, the layer takes inputs from 64 different regions by shifting a filter (weights). The proposed layer is also dissimilar from a fully connected layer since a typical fully connected layer does not share weights while the proposed locally connected layer shares weights between blocks. For all paths in the generator, the first locally connected layer (Loc1) and the second locally connected layer (Loc2) employ the block-size of 1 × 1 and 8 × 8, respectively. The block-size of 8 × 8 is selected considering 8 × 8 block-based inverse DCT in a JPEG decoder. The block-size of 1 × 1 is a special case that can be reproduced by a convolution layer.

Let $\boldsymbol{X}^{\ell}$ and $\boldsymbol{X}^{\ell+1}$ denote the input and the output of the ℓ-th layer in the network. Considering this layer is a locally connected layer, $\boldsymbol{X}^{\ell+1}$ is computed as follows:

$$\begin{aligned}&\boldsymbol{X}^{\ell+1}_{t,b_hp+m,b_wq+n}\\&\quad=\sum_{0\le r<n_r}\sum_{0\le i<b_h}\sum_{0\le j<b_w}\boldsymbol{X}^{\ell}_{r,b_hp+i,b_wq+j}\boldsymbol{W}^{\ell}_{t,r,m,n,i,j}+\boldsymbol{B}^{\ell}_{m,n}\end{aligned}\tag{1}$$

where $\boldsymbol{W}^{\ell}$ and $\boldsymbol{B}^{\ell}$ are the weight and the bias of the layer; b_h and b_w are the height and the width of the block; p and q are the vertical and horizontal block indices; m and n are the vertical and horizontal indices inside of the output block; i and j are the indices inside of the input block; t and r are the indices of the feature maps of $\boldsymbol{X}^{\ell+1}$ and $\boldsymbol{X}^{\ell}$, respectively; n_r is the number of feature maps (channels) in $\boldsymbol{X}^{\ell}$.

In backpropagation, the weights $\boldsymbol{W}$ are updated to minimize loss e using the gradient $\partial e/\partial \boldsymbol{W}$ where e is computed using the objective function in Eq. (10). For the locally connected layer, $\partial e/\partial \boldsymbol{W}^{\ell}$ is computed given the gradient $\partial e/\partial \boldsymbol{X}^{\ell+1}$ from $(\ell+1)$-th layer and $\boldsymbol{W}^{\ell}$. By the chain rule [47]–[49], $\partial e/\partial \boldsymbol{W}^{\ell}$ is represented as follows:

$$\frac{\partial e}{\partial \boldsymbol{W}^{\ell}}=\frac{\partial e}{\partial \boldsymbol{X}^{\ell+1}}\frac{\partial \boldsymbol{X}^{\ell+1}}{\partial \boldsymbol{W}^{\ell}}.\tag{2}$$

For an element $\boldsymbol{W}^{\ell}_{t,r,m,n,i,j}$ in the weight matrix, the gradient of Eq. (2) is expanded as

$$\begin{aligned}&\frac{\partial e}{\partial \boldsymbol{W}^{\ell}_{t,r,m,n,i,j}}\\&\quad=\sum_{0\le p<n_h}\sum_{0\le q<n_w}\frac{\partial e}{\partial \boldsymbol{X}^{\ell+1}_{t,b_hp+m,b_wq+n}}\frac{\partial \boldsymbol{X}^{\ell+1}_{t,b_hp+m,b_wq+n}}{\partial \boldsymbol{W}^{\ell}_{t,r,m,n,i,j}}\\&\quad=\sum_{0\le p<n_h}\sum_{0\le q<n_w}\frac{\partial e}{\partial \boldsymbol{X}^{\ell+1}_{t,b_hp+m,b_wq+n}}\boldsymbol{X}^{\ell}_{r,b_hp+i,b_wq+j}\end{aligned}\tag{3}$$

where n_h and n_w are the number of blocks along the vertical and horizontal axes.

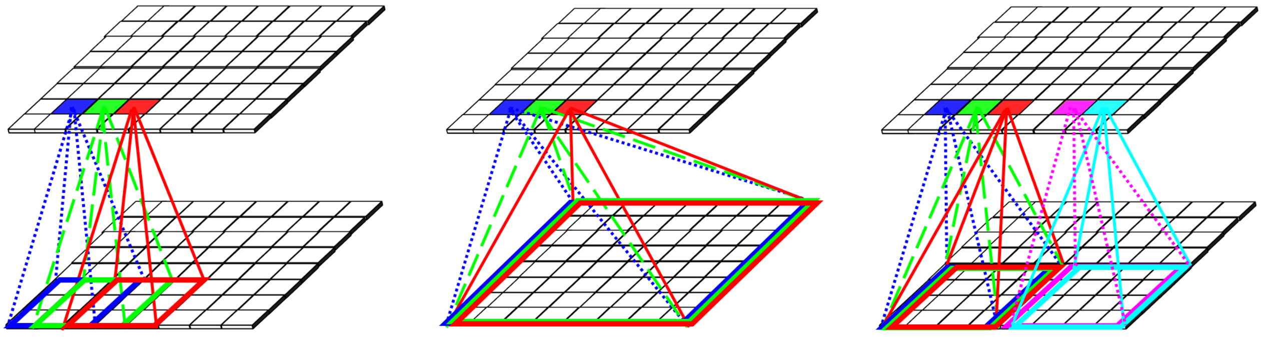
To compute Eq. (2) at the $(\ell-1)$-th layer, the gradient $\partial e/\partial \boldsymbol{X}^{\ell}$ needs to be computed at the ℓ-th layer given the gradient $\partial e/\partial \boldsymbol{X}^{\ell+1}$ from $(\ell+1)$-th layer.

$$\frac{\partial e}{\partial \boldsymbol{X}^{\ell}}=\frac{\partial e}{\partial \boldsymbol{X}^{\ell+1}}\frac{\partial \boldsymbol{X}^{\ell+1}}{\partial \boldsymbol{X}^{\ell}}.\tag{4}$$

For an element $\boldsymbol{X}^{\ell}_{r,b_hp+i,b_wq+j}$, the gradient of Eq. (4) is expanded as

$$\begin{aligned}&\frac{\partial e}{\partial \boldsymbol{X}^{\ell}_{r,b_hp+i,b_wq+j}}\\&\quad=\sum_{t}\sum_{m}\sum_{n}\frac{\partial e}{\partial \boldsymbol{X}^{\ell+1}_{t,b_hp+m,b_wq+n}}\frac{\partial \boldsymbol{X}^{\ell+1}_{t,b_hp+m,b_wq+n}}{\partial \boldsymbol{X}^{\ell}_{r,b_hp+i,b_wq+j}}\\&\quad=\sum_{t}\sum_{m}\sum_{n}\frac{\partial e}{\partial \boldsymbol{X}^{\ell+1}_{t,b_hp+m,b_wq+n}}\boldsymbol{W}^{\ell}_{t,r,m,n,i,j}\end{aligned}\tag{5}$$

where the ranges of summation are $0\le t<n_t$, $0\le m<b_h$, and $0\le n<b_w$; n_t is the number of feature maps (channels) in $\boldsymbol{X}^{\ell+1}$.



(a) Convolution layer

(b) Fully connected layer

(c) Locally connected layer

FIGURE 3. Visual comparison of (a) convolution layer with filter-size of 3 × 3, (b) fully connected layer, and (c) proposed locally connected layer with block-size of 4 × 4. The locally connected layer operates comparable to block-based processing. Each region of output is produced by the summation of the multiplication of the corresponding region of input and shared weights. The weights are shared between blocks, but not between outputs in a block.

Chroma subsampling is processed by averaging the amplitudes of the chrominance component of each block and by subsampling from the block to a scalar. In this article, we investigate the proposed architecture using the popular subsampling ratios, 4:4:4, 4:2:2, and 4:2:0. The 4:4:4 ratio means no subsampling and preserves all the chrominance information. The 4:2:2 mode averages and subsamples with 2:1 ratio for only the horizontal axis. Consequently, the horizontal resolution of the output is half of the input. For the 4:2:0 subsampling, both horizontal and vertical axes are averaged and subsampled with 2:1 ratio. Consequently, each block of 2 × 2 pixels is turned to a scalar (see Figure 2(a)).

Forward processing all the layers in the proposed generator before quantization generates amplitudes of 2D cosine functions for luminance and chrominance components. Quantization is then performed using a conventional quantization method in JPEG compression. We employ the conventional method so that the final output is able to be de-quantized by using a typical JPEG decoder. Quantization is performed by dividing amplitudes by quantization matrices and by rounding the quantized amplitudes to an integer [50], [51]. The gradient of the rounding process is assumed as 1 during training since it is not differentiable. This straight-through estimator approach was analyzed and employed in [52]–[54]. To the best of our knowledge, this is one of the most widely used and reliable approaches. Nevertheless, the assumption is not always correct, and we believe that an enhanced method of handling gradients of rounding functions can improve the performance of the proposed method. The quantization matrices are determined based on user-selected quality factor and can also be selected in an encoding process. Given a trained model, users can control the trade-off between the quality and the size of synthesized images for their applications by varying the quantization matrices. We employ popular quantization matrices that are shown in Eq. (6) and Eq. (7) [55]. The given quantization matrices ($Q_{l,50}$, $Q_{c,50}$) are for the quality factor of 50. The former and the latter matrices are for the luminance component (Q_l) and chrominance part (Q_c). While the matrices have been utilized widely, users can also select other quantization matrices as long as the same matrices are employed in the decoding process. The proposed method is not constrained by certain quantization matrices.

$$Q_{l,50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \tag{6}$$

$$Q_{c,50} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}. \tag{7}$$

Quantization matrix for another quality factor for luminance component is computed as follows:

$$Q_{l,n} = \begin{cases} \max(1, \lfloor \frac{100-n}{50} Q_{l,50} + 0.5 \rfloor), & \text{if } n \geq 50. \\ \lfloor \frac{50}{n} Q_{l,50} + 0.5 \rfloor, & \text{otherwise.} \end{cases} \tag{8}$$

where $n \in (0, 100]$ denotes a quality factor. The conversion of the quantization matrix for chrominance is equivalent.

The architecture in Figure 2 is used for the LSUN bedroom dataset [56] which aims to generate images with the resolution of 64 × 64. For the CIFAR-10 dataset [57] whose objective resolution is 32 × 32, the output dimensions of all the layers in both the generator and the discriminator are reduced by half for both x- and y-axes. Also, the number of activations (feature maps) is reduced by half up to the last residual block in both the generator and the discriminator.

B. DECODER BETWEEN GENERATOR AND DISCRIMINATOR

Since the discriminator takes half of the inputs from the generator and the other half from the training dataset, the two data should be in the same domain (representation). However, the training data set contains real images in the RGB domain, and the outputs of the proposed generator are JPEG compressed data. Hence, we have to either compress the training data or decode the outputs of the generator so that the two images are in the same domain.

We examined both alternatives (see Table 3). It turns out that it is better to decode outputs of the generator before providing them for inputs to the discriminator. Our opinion is that convolution layers, which are major elements of the discriminator, are invented for real images which usually have a continuous tone. However, the compressed data contains amplitudes of block-based DCT which vary largely at the boundary of blocks and also in the blocks. Consequently, the discriminator provides inferior-quality gradients to the generator and hinders training a good generator. Hence, the proposed framework has a decoder that takes outputs of the generator and renders inputs to the discriminator during training. We do not need this conversion (decoder) after training since we only utilize the discriminator for training and our goal is generating compressed data.

Given an output of the generator during training, we first de-quantize the amplitudes by multiplying them by the corresponding quantization matrices used in the generator. We then apply inverse DCT to transform the amplitudes in the frequency domain to the contents in the 2D color domain. We upsample chrominance components to the same resolution of the luminance component if chroma subsampling is applied in the generator. We then convert the amplitudes in the YCbCr space to the RGB space. Lastly, we clip the amplitudes so that after compensating shifting and scaling, the amplitudes are in the range of [0, 255].

C. TRAINING

As GANs are difficult to train [1], many studies have been conducted to improve the stability of training [3], [4], [18]. Still, by employing current state-of-the-art training algorithms to our problem, we had difficulty in training the proposed networks. Hence, we propose a novel loss function to train the proposed framework.

Given a loss function, L, the generator G and the discriminator D are trained by playing a minimax game as follows:

$$\min_{G} \max_{D} L(G, D) \tag{9}$$

Considering the objective function in the Wasserstein GAN with gradient penalty [4], we propose a loss function $L(\cdot)$ by adding an auxiliary loss term $|P(G(\tilde{z}))-\hat{G}(\tilde{z})|$ to guide locally connected layers. By utilizing this function $L(\cdot)$, the generator G learns to synthesize images in a compressed domain where the generated images have similarity with the training data. Also, the generator is overseen to utilize locally connected layers for block-wise processing in image coding/decoding. The function $L(\cdot)$ is defined as follows:

$$\begin{aligned} L(G, D, P) = {} & \mathbb{E}_{\tilde{z}\sim\mathbb{P}_{\ell}}[D(P(G(\tilde{z}))) + \gamma|P(G(\tilde{z})) - \hat{G}(\tilde{z})|] \\ & - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] + \lambda\mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}}D(\hat{\boldsymbol{x}})\|_2 - 1)^2] \end{aligned} \tag{10}$$

where $\mathbb{P}_{\ell}$ and $\mathbb{P}_r$ denote the latent code distribution and the training data distribution. The authors in [4] implicitly defined $\mathbb{P}_{\hat{\boldsymbol{x}}}$ as sampling uniformly along straight lines between pairs of points sampled from $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$. $\hat{G}$ is the layers in the generator G before any locally connected layer. $\hat{G}$ is initialized by the parameters that are trained using [4]. P is the decoder between the generator and the discriminator. γ is the hyperparameter to weight between typical generator loss and the proposed additional generator loss. Gradient penalty coefficient λ is 10 and γ is 0.1 in all experiments. The learning rates for the discriminator and the generator are 0.0003 and 0.0001, respectively.

We believe that further studying on an optimization algorithm for non-typical images can improve the quality of generated results further. However, developing a novel optimization algorithm is beyond the scope of this article.

IV. EXPERIMENTS AND RESULTS

A. DATASET

We experiment using the CIFAR-10 training dataset [57] and the LSUN bedroom training dataset [56]. The CIFAR-10 dataset consists of 50,000 images with the resolution of 32 × 32. The dataset includes images from 10 categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). The LSUN bedroom dataset consists of 3,033,042 bedroom images. The images are scaled to 64 × 64 following the previous work [4].

B. METRIC

We use the Fréchet Inception Distance (FID) [20] which was improved from the Inception score [18] by considering the statistics of real data. The FID computes the Fréchet distance (also known as Wasserstein-2 distance) [58], [59] between the statistics of real data and that of generated samples. The distance is computed using the first two moments (mean and covariance) of activations from the last pooling layer in the Inception v3 model [60]. The FID d is computed as follows:

$$\begin{aligned} d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = {} & ||\boldsymbol{m} - \boldsymbol{m}_w||_2^2 \\ & + \mathrm{Tr}(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}) \end{aligned} \tag{11}$$

where $(\boldsymbol{m}, \boldsymbol{C})$ and $(\boldsymbol{m}_w, \boldsymbol{C}_w)$ represent the mean and covariance of generated samples and of the real dataset. $(\boldsymbol{m}_w, \boldsymbol{C}_w)$

TABLE 1. Quantitative comparison using FIDs and compression ratio for the CIFAR-10 dataset [57]. The lower FID means closer to the statistics of real data. The first and second rows show FIDs of training images and of generated RGB images using the TTUR method [20] by processing compression in a post-processing stage. The last three rows present generating compressed data directly using the TTUR method [20], the FC Generator in the TTUR [20], and the proposed method.

(FID / compression ratio)

Method	Generator output	Chroma subsampling	Quality factor			
			100	75	50	25
Real data	-	4:4:4	0.02 / 3.0:1	6.49 / 16.1:1	16.89 / 23.9:1	35.76 / 34.8:1
		4:2:2	0.68 / 3.3:1	10.59 / 17.8:1	23.89 / 27.0:1	45.40 / 40.4:1
		4:2:0	1.86 / 3.4:1	16.30 / 18.2:1	32.47 / 28.1:1	55.83 / 43.1:1
TTUR [20]	RGB image	4:4:4	26.10 / **2.6:1**	31.77 / 14.8:1	41.02 / 22.6:1	54.21 / 34.1:1
		4:2:2	**25.79 / 3.1:1**	37.03 / 16.5:1	48.45 / 25.5:1	63.37 / 39.6:1
		4:2:0	26.90 / **3.2:1**	43.31 / 17.0:1	56.22 / 26.4:1	72.30 / 41.8:1
TTUR [20]	JPEG compressed image	4:4:4	70.98 / 1.6:1	77.00 / 11.9:1	82.55 / 18.8:1	89.15 / 29.5:1
		4:2:2	70.58 / 2.3:1	80.99 / 14.4:1	87.65 / 22.7:1	94.24 / 36.5:1
		4:2:0	68.04 / 3.0:1	79.92 / **18.1:1**	88.77 / **30.2:1**	97.05 / **50.3:1**
FC generator [20]		4:4:4	84.92 / 1.4:1	89.34 / 9.8:1	93.94 / 16.0:1	96.06 / 26.5:1
		4:2:2	73.85 / 2.2:1	87.07 / 12.8:1	93.31 / 20.6:1	96.13 / 34.2:1
		4:2:0	84.47 / 2.5:1	94.14 / 13.0:1	100.02 / 20.7:1	102.21 / 34.5:1
Proposed method		4:4:4	**25.29** / 2.4:1	**31.74 / 15.1:1**	**40.49 / 22.9:1**	**53.60 / 34.5:1**
		4:2:2	25.80 / 2.9:1	**36.81 / 16.7:1**	**47.69 / 25.8:1**	**62.46 / 40.0:1**
		4:2:0	**26.78 / 3.2:1**	**43.18** / 17.1:1	**55.67** / 26.7:1	**71.48** / 42.2:1

are measured using the entire images in the training dataset. $(\boldsymbol{m}, \boldsymbol{C})$ are computed using 50,000 generated images.

While PSNR and MS-SSIM are more widely-used metrics to measure image qualities, the metrics are full-reference metrics. They consequently require reference (ground truth) images to measure the qualities. However, synthesized images using the proposed method do not have any reference images. Hence, we use FID to evaluate synthesized images.

C. RESULTS

We analyze the proposed method and the architectures in the TTUR method [20] by training them using the datasets in Section IV-A and by evaluating them quantitatively and qualitatively. For quantitative comparison, we measure the Fréchet Inception Distance (FID) in Section IV-B. Table 1 and Table 2 show the quantitative results for the CIFAR-10 dataset [57] and the LSUN bedroom dataset [56], respectively. In both tables, we show the FIDs for three chroma subsampling ratios (4:4:4, 4:2:2, 4:2:0) and for four quality factors of quantization (100, 75, 50, 25). The first row shows the FID variations of real images by applying chroma subsampling and quantization. The second row presents the FID of the original TTUR method generating RGB images [20]. For this analysis, JPEG compression is processed as a post-processing step that follows the neural networks. The third row shows the result of the TTUR method for generating JPEG compressed images directly. The fourth row presents the result of the fully connected (FC) generator in the TTUR method for generating JPEG compressed images directly. In the last row, we show the result of the proposed method.

We show visual results of the CIFAR-10 and the LSUN bedroom datasets in Figures 4 and 5, respectively. On the left side, we denote the quality factor for quantization and chroma subsampling ratios. We show the results of (100, 4:4:4), (100, 4:2:2), (100, 4:2:0), (75, 4:4:4), (50, 4:4:4), and (25, 4:4:4) from the first row to the last row. On the first column, we show the results of real images that are processed by the corresponding compression. The second column shows the results of the generated images using the original TTUR method [20]. The result images are first generated as RGB images and are then coded and decoded in a post-processing stage. The third column shows the result of the TTUR method [20], and the fourth column presents the result of the FC generator in the TTUR method [20]. The last column shows the results of the proposed method. The last three columns are the results of generating JPEG compressed images in the networks.

To compute the FID in the first row in both tables, we first encode and decode 50,000 training images and then compute the statistics of the processed images. We then estimate the FID between the statistics of the original training data and those of the processed images. For the CIFAR-10 dataset in Table 1, the distance is close to 0 using the chroma subsampling ratio of 4:4:4 and the quality factor of 100. It's quite small since the encoding/decoding does not impact the images much (only small rounding errors, etc). By decreasing quality factor and by subsampling from a larger region, the encoding/decoding distorts images more and hence, FID increases. Since decreasing quality factor by 25 affects images much more than adjusting chroma subsampling from 4:4:4 to 4:2:0, FID is also increased by a larger amount.

TABLE 2. Quantitative comparison using FIDs and compression ratio for the LSUN dataset [56]. The lower FID means closer to the statistics of real data. The first and second rows show FIDs of training images and of generated RGB images using the TTUR method [20] by processing compression in a post-processing stage. The last three rows present generating compressed data directly using the TTUR method [20], the FC Generator in the TTUR [20], and the proposed method.

(FID / compression ratio)

Method	Generator output	Chroma subsampling	Quality factor			
			100	75	50	25
Real data	-	4:4:4	9.96 / 4.1:1	6.82 / 20.4:1	7.57 / 30.7:1	18.26 / 43.7:1
		4:2:2	10.36 / 5.2:1	7.13 / 23.7:1	8.18 / 35.7:1	18.94 / 52.3:1
		4:2:0	11.37 / 6.0:1	10.22 / 25.2:1	12.31 / 38.6:1	23.01 / 58.0:1
TTUR [20]	RGB image	4:4:4	12.44 / **3.9:1**	10.24 / **21.4:1**	11.35 / 32.3:1	**23.17** / 45.9:1
		4:2:2	13.41 / **4.8:1**	12.05 / **24.3:1**	13.34 / **37.3:1**	24.46 / **55.0:1**
		4:2:0	14.80 / **4.6:1**	16.00 / **25.7:1**	17.98 / 40.1:1	27.99 / 60.9:1
TTUR [20]	JPEG compressed image	4:4:4	65.14 / 2.5:1	61.38 / 20.2:1	70.54 / **32.6:1**	95.33 / **47.9:1**
		4:2:2	88.51 / 3.2:1	95.95 / 22.1:1	103.72 / 35.3:1	121.48 / 53.8:1
		4:2:0	93.78 / 3.5:1	104.25 / 23.6:1	112.71 / 38.9:1	129.77 / 60.8:1
FC generator [20]		4:4:4	138.53 / 1.8:1	138.00 / 12.7:1	146.74 / 21.9:1	162.30 / 38.5:1
		4:2:2	163.10 / 2.2:1	157.31 / 14.2:1	160.45 / 24.5:1	168.16 / 42.5:1
		4:2:0	171.27 / 2.8:1	171.12 / 15.7:1	171.14 / 26.6:1	173.80 / 46.7:1
Proposed method		4:4:4	**12.13** / 3.3:1	**9.99 / 21.4:1**	**11.25** / 32.3:1	23.18 / 45.9:1
		4:2:2	**12.96** / 4.0:1	**11.57 / 24.3:1**	**13.12 / 37.3:1**	**24.28** / 54.9:1
		4:2:0	**14.21** / 4.1:1	**15.41 / 25.7:1**	**17.63 / 40.2:1**	**27.82 / 61.0:1**

TABLE 3. Analysis on the input data domain for discriminator using the CIFAR-10 dataset [57]. The lower FID means the better result. The top row shows FIDs when the model is trained using compressed images. The bottom row shows FIDs where the model is trained using RGB images.

(FID / compression ratio)

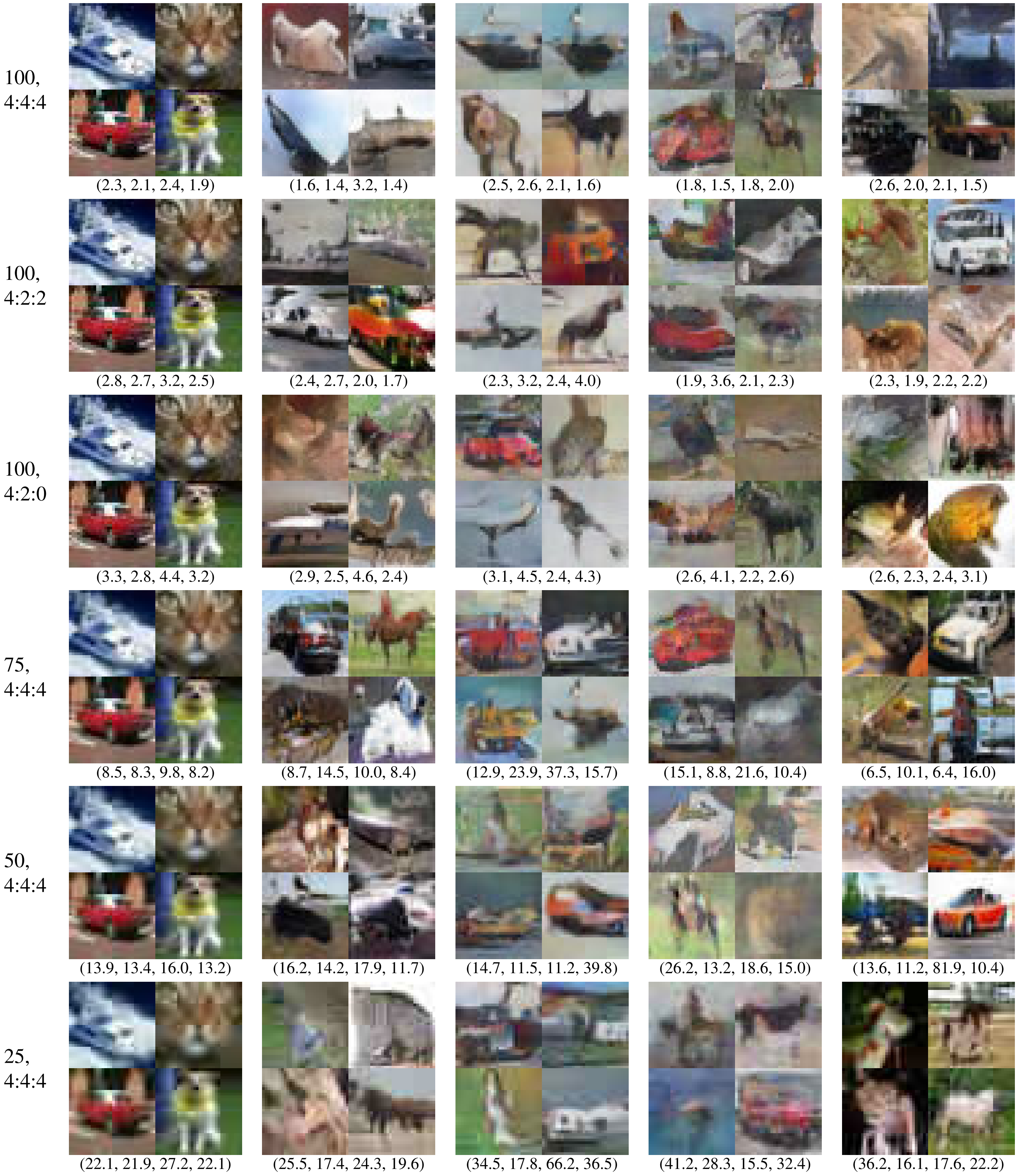
Input data domain for discriminator	Generator output	Chroma subsampling	Quality factor			
			100	75	50	25
Compressed domain	JPEG compressed image	4:4:4	247.21 / 1.9:1	163.60 / **17.1:1**	142.69 / **26.2:1**	150.13 / **37.8:1**
		4:2:2	173.03 / 2.0:1	131.94 / 15.4:1	130.35 / 22.9:1	125.58 / 38.9:1
		4:2:0	176.29 / 2.2:1	124.29 / 15.1:1	130.41 / **26.9:1**	127.58 / **47.8:1**
RGB domain		4:4:4	**25.29 / 2.4:1**	**31.74** / 15.1:1	**40.49** / 22.9:1	**53.60** / 34.5:1
		4:2:2	**25.80 / 2.9:1**	**36.81 / 16.7:1**	**47.69 / 25.8:1**	**62.46 / 40.0:1**
		4:2:0	**26.78 / 3.2:1**	**43.18 / 17.1:1**	**55.67** / 26.7:1	**71.48** / 42.2:1

TABLE 4. Analysis on the proposed objective function using the CIFAR-10 dataset [57]. The lower FID means the better result. The top row and bottom row present FIDs of generated images using WGAN-GP loss in [4] and the proposed loss function in Eq. (10), respectively.

(FID / compression ratio)

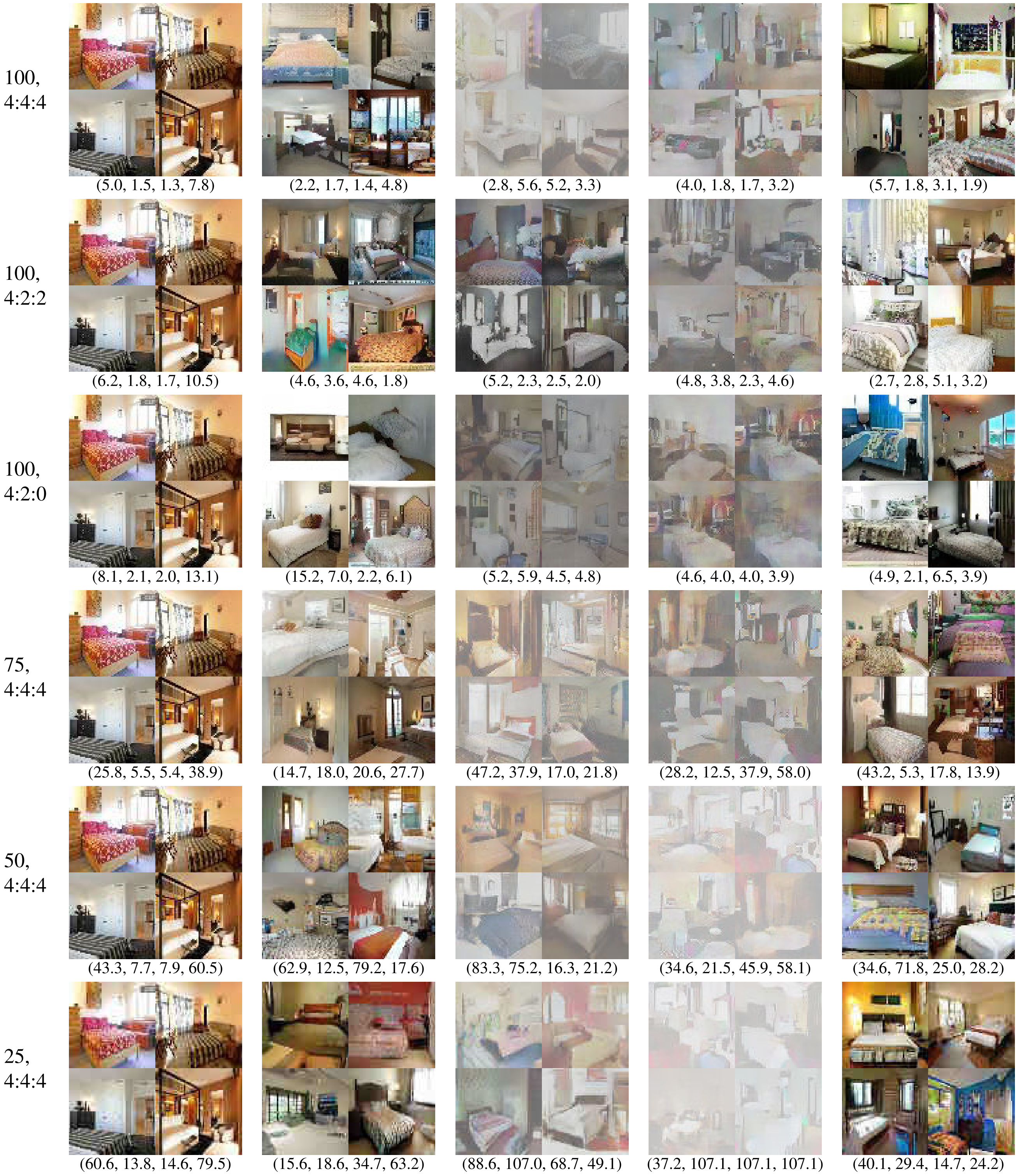
Objective function	Generator output	Chroma subsampling	Quality factor			
			100	75	50	25
WGAN-GP loss in [4]	JPEG compressed image	4:4:4	36.85 / **3.4:1**	39.76 / **16.1:1**	46.50 / **24.3:1**	57.34 / **36.1:1**
		4:2:2	49.11 / **3.2:1**	58.83 / **16.7:1**	67.47 / **26.1:1**	77.87 / **40.3:1**
		4:2:0	51.30 / **3.2:1**	65.88 / 16.8:1	75.01 / 26.3:1	82.86 / 40.7:1
Proposed loss in Eq. (10)		4:4:4	**25.29** / 2.4:1	**31.74** / 15.1:1	**40.49** / 22.9:1	**53.60** / 34.5:1
		4:2:2	**25.80** / 2.9:1	**36.81 / 16.7:1**	**47.69** / 25.8:1	**62.46** / 40.0:1
		4:2:0	**26.78 / 3.2:1**	**43.18 / 17.1:1**	**55.67 / 26.7:1**	**71.48 / 42.2:1**

For the LSUN bedroom dataset in Table 2, the distance of real data using the chroma subsampling ratio of 4:4:4 and the quality factor of 100 is much greater than that in the CIFAR-10 dataset. The FID is defined by the distance between the statistics of the entire training data and those of 50,000 processed or generated images. Since the



(a) Real data (b) TTUR (RGB) (c) TTUR (encoded) (d) FC generator (encoded) (e) Proposed method

FIGURE 4. Visual results of generating compressed images using the CIFAR-10 dataset [57]. On the left side, we denote the quality factor for quantization and chroma subsampling ratios. We show the results of (100, 4:4:4), (100, 4:2:2), (100, 4:2:0), (75, 4:4:4), (50, 4:4:4), and (25, 4:4:4) from the first row to the last row. The first and second columns show the results of real images and the original TTUR method [20] that are processed by the corresponding encoding/decoding. The third and fourth columns show the results of the TTUR method and the FC generator in the TTUR method that generates compressed data directly. The last column shows the result of the proposed method. We also present the compression ratio for each image at the bottom of each set of images. The ratio is written in the clockwise order from the top-left image.



(a) Real data (b) TTUR (RGB) (c) TTUR (encoded) (d) FC generator (encoded) (e) Proposed method

FIGURE 5. Visual results of generating compressed images using the LSUN dataset [56]. On the left side, we denote the quality factor for quantization and chroma subsampling ratios. We show the results of (100, 4:4:4), (100, 4:2:2), (100, 4:2:0), (75, 4:4:4), (50, 4:4:4), and (25, 4:4:4) from the first row to the last row. The first and second columns show the results of real images and the original TTUR method [20] that are processed by the corresponding encoding/decoding. The third and fourth columns show the results of the TTUR method and the FC generator in the TTUR method that generates compressed data directly. The last column shows the result of the proposed method. We also present the compression ratio for each image at the bottom of each set of images. The ratio is written in the clockwise order from the top-left image.

TABLE 5. Analysis on the proposed locally connected layer using the CIFAR-10 dataset [57]. The lower FID means the better result. The first, second, and third row present the FIDs of generated images using convolution layers, fully connected layers, and locally connected layers, respectively.

(FID / compression ratio / ms)

Objective function	Generator output	Chroma subsampling	Quality factor			
			100	75	50	25
Convolution layer	JPEG compressed image	4:4:4	102.56 / **2.4:1 / 7.3**	97.94 / 13.0:1 / 3.8	102.61 / 20.3:1 / 3.5	103.55 / 32.2:1 / 3.2
		4:2:2	132.75 / 2.4:1 / **6.5**	119.81 / 14.4:1 / **3.6**	125.55 / 22.7:1 / **3.2**	141.56 / 36.5:1 / **3.1**
		4:2:0	146.05 / 2.8:1 / **5.7**	120.83 / 15.1:1 / **3.2**	126.14 / 23.8:1 / **3.0**	153.39 / 38.7:1 / **2.9**
Fully connected layer		4:4:4	26.95 / 2.0:1 / 7.4	35.13 / 14.1:1 / **3.7**	43.77 / 22.6:1 / **3.4**	55.83 / **34.6:1 / 3.1**
		4:2:2	29.35 / 2.6:1 / **6.5**	39.55 / 16.1:1 / **3.6**	50.15 / 25.7:1 / 3.3	64.30 / **40.0:1** / 3.2
		4:2:0	32.92 / 3.0:1 / **5.7**	46.84 / 16.4:1 / 3.3	59.10 / 26.5:1 / **3.0**	73.90 / **42.4:1 / 2.9**
Locally connected layer		4:4:4	**25.29 / 2.4:1** / 7.4	**31.74 / 15.1:1** / 3.8	**40.49 / 22.9:1** / 3.5	**53.60** / 34.5:1 / 3.2
		4:2:2	**25.80 / 2.9:1** / 6.6	**36.81 / 16.7:1 / 3.6**	**47.69 / 25.8:1** / 3.3	**62.46 / 40.0:1 / 3.1**
		4:2:0	**26.78 / 3.2:1** / 5.8	**43.18 / 17.1:1** / 3.3	**55.67 / 26.7:1 / 3.0**	**71.48** / 42.2:1 / 3.0

number of images in the CIFAR-10 dataset is 50,000, the distance is quite small considering the encoding/decoding does not distort much. However, since the LSUN bedroom dataset contains 3,033,042 images, 50,000 images should be sampled to compute the statistics of the processed images. It causes a relatively larger FID for the LSUN bedroom dataset. It is also interesting to note that for the LSUN dataset, the quality factor of 75 is better than that of 100 in most experiments. We believe since the bedroom images often have continuous tone because of its contents or preprocessing, discarding high-frequency components decreases FID.

FIDs in the second row is computed by first generating RGB images using the TTUR method [20] and by encoding/decoding the generated RGB images. As generating RGB images have been studied a lot in recent years and the TTUR method is one of the state-of-the-art methods, generated RGB images are quite visually plausible. FID is increased by encoding/decoding the images using a lower quality factor and subsampling from a larger region. Some of the distortions can be visually observed in Figs. 4 and 5.

The third row presents applying the same method to generate JPEG encoded images. The results demonstrate that directly applying the method does not produce competitive results. The fourth row shows the results of applying the FC generator in the TTUR method [20] for generating encoded images. While FC generator often performs poorer than the selected TTUR method for generating typical images, we tried the FC generator to avoid extensively applied convolution layers in the TTUR method. However, the FC generator does not perform well even for generating encoded images.

The last row in both tables shows the results of the proposed method. The proposed method achieves promising results for generating JPEG encoded images directly. The proposed method outperforms applying the TTUR method for generating JPEG encoded image directly. Moreover, the proposed method is competitive to the method that generates RGB images using the TTUR method and compresses them by post-processing.

D. ANALYSIS

To analyze statistical significance, we computed the FIDs of all the methods in Table 1 for 100 times with the chroma subsampling ratio of 4:4:4 and the quality factor of 100. Each FID is computed using 50,000 synthesized images as mentioned in Section IV-B. Considering synthesizing compressed data directly, the FIDs of the proposed method (last row) were always lower than those of the TTUR method (third row) [20] and those of the FC Generator in the TTUR (fourth row) [20]. Comparing the FIDs of the proposed method (synthesizing compressed data directly) to those of the TTUR method [20] (compressing them in post-processing), the former (last row) was also always lower than the latter (second row).

As mentioned in Section III-B, we need to make the inputs to the discriminator in the same domain by either compressing the training data or decoding the outputs of the generator. We present an ablation study on the two cases in Table 3 using the CIFAR-10 dataset [57]. The inputs to the discriminator are in the compressed domain and in the RGB domain for the top row and the bottom row, respectively. In other words, the top row shows the results of compressing the training data, and the bottom row presents the results of decoding the outputs of the generator. Table 3 shows that decoding the outputs of the generator is better than compressing the training data. Hence, the proposed framework has a decoder that takes outputs of the generator and renders inputs to the discriminator during training.

To demonstrate the effectiveness of the proposed loss function, we present an ablation study in Table 4 using the CIFAR-10 dataset [57]. The top row shows the results using the objective function in [4], and the bottom row presents those using the proposed loss function. Table 4 demonstrates that the model trained using the proposed loss function achieves lower FID (better quality) while having a competitive compression ratio.

To analyze the proposed locally connected layers, we show an ablation study in Table 5 using the CIFAR-10 dataset [57]. The top row and the second row show the results of the architectures that are based on the proposed networks, but replaced the locally connected layers by convolution layers and by fully connected layers, respectively. The results show that the

TABLE 6. **Analysis on the processing time of synthesizing compressed image data from noise using the CIFAR-10 dataset [57].**

(ms)

Method	Generator output	Chroma subsampling	Quality factor			
			100	75	50	25
TTUR [20]	RGB image	4:4:4	8.2	4.5	4.1	3.8
		4:2:2	7.3	4.3	4.0	3.8
		4:2:0	6.5	3.9	3.7	3.6
TTUR [20]	JPEG compressed image	4:4:4	7.4	3.7	3.3	3.0
		4:2:2	6.6	3.5	3.2	3.0
		4:2:0	5.8	3.2	2.9	2.8
FC generator [20]		4:4:4	6.8	3.2	2.8	2.6
		4:2:2	6.0	3.0	2.7	2.5
		4:2:0	5.2	2.7	2.4	2.4
Proposed method		4:4:4	7.4	3.8	3.5	3.2
		4:2:2	6.6	3.6	3.3	3.1
		4:2:0	5.8	3.3	3.0	3.0

TABLE 7. **Analysis on the hyperparameters in the proposed objective function using the CIFAR-10 dataset [57]. The generator output is a JPEG compressed image, and the chroma subsampling is 4:4:4. The lower FID means the better result.**

Hyperparameters	Quality factor			
(λ, γ)	100	75	50	25
(1, 0.1)	25.49	31.80	40.67	53.87
(10, 0.01)	25.49	31.75	40.87	53.61
(10, 0.1)	**25.29**	**31.74**	**40.49**	**53.60**
(10, 1)	25.50	31.88	40.56	53.95
(100, 0.1)	25.40	32.05	40.70	53.98

FIDs of the network with locally connected layers are lower than those of the networks with convolution layers and with fully connected layers. The network with convolution layers is inferior to the others since convolution layers repeatedly utilize the same weights and biases at all output locations while block-based operations in JPEG demand quite different values at each location (both within a block and between blocks).

To present the efficiency of the proposed method, we show the processing time of synthesizing compressed image data from noise in Table 6. The proposed method (last row) takes less time comparing to synthesizing RGB images and compressing them in a separate step (first row) while the two methods have competitive FIDs and compression ratios as shown in Tables 1 and 2. The proposed method takes longer time comparing to synthesizing compressed image data directly using the TTUR method [20] and the FC generator in [20] since the proposed method has additional layers such as the proposed locally connected layers. However, the visual qualities and FIDs of the proposed method outperform two methods (second and third rows) (see Figures 4 and 5 and Tables 1 and 2). The processing time was measured using a computer with an Intel Xeon Processor (E5-2609 v4) and an NVIDIA GeForce RTX 2080 Ti GPU. We generated 50,000 images using the batch size of 1,000, then calculated average processing time per image.

Table 7 shows the analysis of the hyperparameters in the proposed objective function. We trained the proposed architecture with varying hyperparameters and generated 50,000 images using each trained model. We then measured FIDs between the statistics of the generated images and those of the original training data.

V. CONCLUSION

We presented a generative adversarial network framework that directly generates images in the compressed domain rather than generating raw RGB images. We proposed a novel generator consisting of the proposed locally connected layers, chroma subsampling layers, quantization layers, residual blocks, and convolution layers. We also presented training strategies for the proposed framework including the loss function and the decoder between the generator and the discriminator. We demonstrated that the proposed framework outperforms applying the state-of-the-art GANs for generating compressed data directly. Moreover, we showed that the proposed method achieves competitive results comparing to generating raw RGB images using one of the state-of-the-art methods and compressing the images by post-processing. We believe that the proposed method can be further improved by investigating optimization algorithms for learning to generate compressed data. We also believe that if quality-enhanced GANs are later available, the proposed method can be extended and applied to the networks to synthesize higher-quality images in the compressed domain.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015. [Online]. Available: http://arxiv.org/abs/1511.06434

[3] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, D. Precup and Y. W. Teh, Eds. Sydney, NSW, Australia: International Convention Centre, 2017, pp. 214–223.

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 5767–5777. [Online]. Available: http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf

[5] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 613–621. [Online]. Available: http://papers.nips.cc/paper/6194-generating-videos-with-scene-dynamics.pdf

[6] W. Xiong, W. Luo, L. Ma, W. Liu, and J. Luo, "Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2364–2373.

[7] L. Sixt, B. Wild, and T. Landgraf, "RenderGAN: Generating realistic labeled data," *Frontiers Robot. AI*, vol. 5, p. 66, Jun. 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2018.00066

[8] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8798–8807.

[9] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, "Video-to-video synthesis," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 1144–1156.

[10] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using GAN for improved liver lesion classification," in *Proc. IEEE 15th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2018, pp. 289–293.

[11] *Encoding Parameters of Digital Television for Studios*, Standard ITU-R Recommendation 601, International Telecommunication Union, 1982.

[12] *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Requirements and Guidelines*, Standard ITU-T T.81, International Telecommunication Union, 1992.

[13] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[14] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[15] G. Hudson, A. Leger, B. Niss, and I. Sebestyen, "JPEG at 25: Still going strong," *IEEE Multimedia Mag.*, vol. 24, no. 2, pp. 96–103, Apr. 2017.

[16] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[17] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[18] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 2234–2242. [Online]. Available: http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf

[19] H. L. Prasad, P. La, and S. Bhatnagar, "Two-timescale algorithms for learning Nash equilibria in general-sum stochastic games," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.* Richland, SC, USA: International Foundation Autonomous Agents Multiagent Systems, 2015, pp. 1371–1379. [Online]. Available: http://dl.acm.org/citation.cfm?id=2772879.2773328

[20] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 6626–6637. [Online]. Available: http://papers.nips.cc/paper/7240-gans-trained-by-a-two-time-scale-update-rule-converge-to-a-local-nash-equilibrium.pdf

[21] L. Theis, W. Shi, A. Cunningham, and F. Huszar, "Lossy image compression with compressive autoencoder," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–19.

[22] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–12. [Online]. Available: http://arxiv.org/abs/1511.06085

[23] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5435–5443.

[24] S. Santurkar, D. Budden, and N. Shavit, "Generative compression," in *Proc. Picture Coding Symp. (PCS)*, Jun. 2018, pp. 258–262.

[25] T. Kaneko, S. Takaki, H. Kameoka, and J. Yamagishi, "Generative adversarial network-based postfilter for STFT spectrograms," in *Proc. Interspeech*, Aug. 2017, pp. 3389–3393, doi: 10.21437/Interspeech.2017-962.

[26] M. Mardani, E. Gong, J. Y. Cheng, S. S. Vasanawala, G. Zaharchuk, L. Xing, and J. M. Pauly, "Deep generative adversarial neural networks for compressive sensing MRI," *IEEE Trans. Med. Imag.*, vol. 38, no. 1, pp. 167–179, Jan. 2019.

[27] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 105–114.

[28] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Computer Vision—ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham, Switzerland: Springer, 2019, pp. 63–79.

[29] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using Cycle-in-Cycle generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 81–814.

[30] A. Lucas, S. Lopez-Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3312–3327, Jul. 2019.

[31] 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. (CVPR)*, Jul. 2017, pp. 5967–5976.

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

[33] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 700–708. [Online]. Available: http://papers.nips.cc/paper/6672-unsupervised-image-to-image-translation-networks.pdf

[34] J.-Y. 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. 2242–2251.

[35] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5505–5514.

[36] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6882–6890.

[37] Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia, "Image inpainting via generative multi-column convolutional neural networks," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, K. Grauman, H. Larochelle, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 331–340. [Online]. Available: http://papers.nips.cc/paper/7316-image-inpainting-via-generative-multi-column-convolutional-neural-networks.pdf

[38] S. Tripathi, Z. C. Lipton, and T. Q. Nguyen, "Correction by projection: Denoising images with generative adversarial networks," *CoRR*, vol. abs/1803.04477, 2018. [Online]. Available: http://arxiv.org/abs/1803.04477

[39] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun, and G. Wang, "Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1348–1357, Jun. 2018.

[40] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, M. F. Balcan and K. Q. Weinberger, Eds. New York, NY, USA: PMLR, Jun. 2016, pp. 1060–1069. [Online]. Available: http://proceedings.mlr.press/v48/reed16.html

[41] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5908–5916.

[42] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN++: Realistic image synthesis with stacked generative adversarial networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1947–1962, Aug. 2019.

[43] L. Galteri, L. Seidenari, M. Bertini, and A. D. Bimbo, "Deep generative adversarial compression artifact removal," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4836–4845.

[44] L. Galteri, L. Seidenari, M. Bertini, and A. D. Bimbo, "Deep universal generative adversarial compression artifact removal," *IEEE Trans. Multimedia*, vol. 21, no. 8, pp. 2131–2145, Aug. 2019.

[45] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 469–477. [Online]. Available: http://papers.nips.cc/paper/6544-coupled-generative-adversarial-networks.pdf

[46] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–14, Jul. 2017, doi: 10.1145/3072959.3073659.

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[48] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[49] T. Apostol, *Mathematical Analysis* (Addison-Wesley Series in Mathematics). Reading, MA, USA: Addison-Wesley, 1974.

[50] W.-H. Chen and W. Pratt, "Scene adaptive coder," *IEEE Trans. Commun.*, vol. 32, no. 3, pp. 225–232, Mar. 1984.

[51] H. Lohscheller, "A subjectively adapted image communication system," *IEEE Trans. Commun.*, vol. 32, no. 12, pp. 1316–1322, Dec. 1984.

[52] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013. [Online]. Available: http://arxiv.org/abs/1308.3432

[53] M. S. Park, X. Xu, and C. Brick, "Squantizer: Simultaneous learning for both sparse and low-precision neural networks," *CoRR*, vol. abs/1812.08301, 2018. [Online]. Available: http://arxiv.org/abs/1812.08301

[54] P. Yin, J. Lyu, S. Zhang, S. J. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," *CoRR*, vol. abs/1903.05662, 2019. [Online]. Available: http://arxiv.org/abs/1903.05662

[55] H. A. Peterson, H. Peng, J. H. Morgan, and W. B. Pennebaker, "Quantization of color image components in the DCT domain," in *Proc. Hum. Vis., Vis. Process., Digit. Display II*, vol. 13, Jun. 1991, pp. 1453–1453, doi: 10.1117/12.44357.

[56] 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*. [Online]. Available: http://arxiv.org/abs/1506.03365

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

[58] M. Fréchet, "Sur la distance de deux lois de probabilité," *C. R. Acad. Sci. Paris*, vol. 244, pp. 689–692, 1957.

[59] D. C. Dowson and B. V. Landau, "The Fréchet distance between multivariate normal distributions," *J. Multivariate Anal.*, vol. 12, no. 3, pp. 450–455, Sep. 1982. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0047259X8290077X

[60] 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.

BYEONGKEUN KANG received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2013, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California at San Diego, La Jolla, CA, USA, in 2015 and 2018, respectively. He was a Postdoctoral Fellow with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, from 2018 to 2019. He is currently an Assistant Professor with the Seoul National University of Science and Technology, Seoul. His current research interests include semantic segmentation, object detection, and human–machine interaction.

SUBARNA TRIPATHI received the M.S. degree from IIT Delhi, in 2011, and the Ph.D. degree in electrical and computer engineering from the University of California at San Diego, La Jolla, CA, USA, in 2018. She is currently a Researcher at Intel Labs, where she is working in computer vision and machine learning with a focus in scene graphs, graph embeddings, and scene parsing.

TRUONG Q. NGUYEN (Fellow, IEEE) is currently a Professor with the Department of Electrical and Computer Engineering (ECE), University of California at San Diego (UC San Diego). He is a coauthor (with Prof. Gilbert Strang) of a popular textbook, *Wavelets and Filter Banks* (Wellesley-Cambridge Press, 1997), and the author of several MATLAB-based toolboxes on image compression, electrocardiogram compression, and filter bank design. He has over 400 publications. His current research interests include 3-D video processing and communications and their efficient implementation.

He received the IEEE Transactions on Signal Processing Paper Award (image and multidimensional processing area) for the paper he cowrote with Prof. P. P. Vaidyanathan on linear-phase perfect-reconstruction filter banks, in 1992. He also received the NSF Career Award in 1995. He is also the Series Editor of *Digital Signal Processing* (Academic Press). He served as an Associate Editor for the IEEE Transactions on Signal Processing, from 1994 to 1996; IEEE Signal Processing Letters, from 2001 to 2003; the IEEE Transactions on Circuits and Systems, from 1996 to 1997 and from 2001 to 2004; and the IEEE Transactions on Image Processing, from 2004 to 2005.