

# Gated-GAN: Adversarial Gated Networks for Multi-Collection Style Transfer

Xinyuan Chen, Chang Xu, Xiaokang Yang, *Senior Member, IEEE*, Li Song, and Dacheng Tao, *Fellow, IEEE*

arXiv:1904.02296v1 [cs.CV] 4 Apr 2019

**Abstract**—Style transfer describes the rendering of an image’s semantic content as different artistic styles. Recently, generative adversarial networks (GANs) have emerged as an effective approach in style transfer by adversarially training the generator to synthesize convincing counterfeits. However, traditional GAN suffers from the mode collapse issue, resulting in unstable training and making style transfer quality difficult to guarantee. In addition, the GAN generator is only compatible with one style, so a series of GANs must be trained to provide users with choices to transfer more than one kind of style. In this paper, we focus on tackling these challenges and limitations to improve style transfer. We propose adversarial gated networks (Gated-GAN) to transfer multiple styles in a single model. The generative networks have three modules: an encoder, a gated transformer, and a decoder. Different styles can be achieved by passing input images through different branches of the gated transformer. To stabilize training, the encoder and decoder are combined as an auto-encoder to reconstruct the input images. The discriminative networks are used to distinguish whether the input image is a stylized or genuine image. An auxiliary classifier is used to recognize the style categories of transferred images, thereby helping the generative networks generate images in multiple styles. In addition, Gated-GAN makes it possible to explore a new style by investigating styles learned from artists or genres. Our extensive experiments demonstrate the stability and effectiveness of the proposed model for multi-style transfer.

**Index Terms**—Multi-Style Transfer, Adversarial Generative Networks.

## I. INTRODUCTION

STYLE transfer refers to redrawing an image by imitating another artistic style. Specifically, given a reference style, one can make the input image look like it has been redrawn with a different stroke, perceptual representation,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2018.2869695, IEEE Transactions on Image Processing

©20XX IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

X. Chen is with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the Centre for Artificial Intelligence and the Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: xychen91@sjtu.edu.cn).

C. Xu and D. Tao are with the UBTech Sydney Artificial Intelligence Centre and the School of Information Technologies, the Faculty of Engineering and Information Technologies, The University of Sydney, 6 Cleveland St, Darlington, NSW 2008, Australia (e-mail: c.xu@sydney.edu.au; dacheng.tao@sydney.edu.au).

X. Yang and S. Li are with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xkyang@sjtu.edu.cn; song\_li@sjtu.edu.cn).

color scheme, or that it has been retouched using a different artistic interpretation. Manually transferring the image style by a professional artist usually takes considerable time. However, style transfer is a valuable technique with many practical applications, for example quickly creating cartoon scenes from landscapes or city photographs and providing amateur artists with guidelines for painting. Therefore, optimizing style transfer is a valuable pursuit.

Style transfer, as an extension of texture transfer, has a rich history. Texture transfer aims to render an object with the texture extracted from a different object [1], [2], [3], [4]. In the early days, texture transfer used low-level visual features of target images, while the latest style transfer approaches are based on semantic features derived from pre-trained convolutional neural networks (CNNs). Gatys *et al.* [5] introduced the neural style transfer algorithm to separate natural image content and style to produce new images by combining the content of an arbitrary photograph with the styles of numerous well-known works of art. A number of variants emerged to improve the speed, flexibility, and quality of style transfer. Johnson *et al.* [6] and Ulyanov *et al.* [7] accelerated style transfer by using feedforward networks, while Chen *et al.* [8], Li *et al.* [9] and Odena *et al.* [10] achieved multi-style transfer by extracting each style from a single image. Ulyanov *et al.* [11] and Luan *et al.* [12] enhanced the quality of style transfer by investigating instance normalization in feedforward networks [7].

CNN-based style transfer methods can now produce high-quality imitative images. However, these methods focus on transferring the original image to the style provided by another style image (typically a painting). In contrast, collection style transfer aims to stylize a photograph by mimicking an artist’s or genre’s style. In practice, when a user takes a picture of a beautiful landscape, he might hope to re-render it on canvas such that it appears to have been painted by an artist, e.g., *Monet*, or in the style of a famous animation, e.g., *Your Name*. Given an in-depth understanding of an artist’s collection of paintings, it is possible to imagine how the artist might render the scene.

With this in mind, generative adversarial networks (GANs) [13] can be applied to learn the distribution of an artist’s paintings. GANs are a framework in which two neural networks compete with each other: a generative network and a discriminative network. The generative and discriminative networks are simultaneously optimized in a two-player game, where the discriminative networks aim to determine whether or not the input is painted by the artist, while the generative networks learn to generate images to fool the discriminative

networks. However, the GAN training procedure is unstable. In particular, without paired training samples, the original GANs cannot guarantee that the output imitations contain the same semantic information as that of the input images. CycleGAN [14], DiscoGAN [15], DualGAN [16] proposed cycle-consistent adversarial networks to address the unpaired image-to-image translation problem. They simultaneously trained two pairs of generative networks and discriminative networks, one to produce imitative paintings and the other to transform the imitation back to the original photograph and pursue cycle consistency.

Considering the wide application of style transfer on mobile devices, space-saving is an important algorithm design consideration. Methods of CycleGAN [14], DiscoGAN [15], DualGAN [16] could only transfer one style per network. In this work, we propose a gated transformer module to achieve multi-collection style transfer in a single network. Moreover, previous methods adopted cycle-consistent loss requires an additional network that converts the stylized image into the original one. With the increase of the number of transferred style, the training algorithm will become complicated if we adopt cycle-consistent loss. Also, style transfer is actually a one-sided translation problem, which does not expect style images to be transformed to content images. In our method, we adopt encoder-decoder subnetwork and an auto-encoder reconstruction loss to guarantee that the outputs have the consistent semantic information with the content images. With auto-encoder reconstruction loss, our algorithm achieves one-sided mapping, which needs less parameters and can be easily generalized for multiple styles.

The proposed adversarial gated networks (Gated-GAN) realize the transfer of multiple artist or genre styles in a single network (see Figure 1). Different to the conventional encoder-decoder architectures in [6], [17], [14], we additionally consider a gated-transformer network between the encoder and decoder consisting of multiple gates, each corresponding to one style. The gate controls which transformer is connected to the model so that users can switch gate to choose between different styles. If the gated transformer is skipped, the encoder and decoder are trained as an auto-encoder to preserve semantic consistency between input images and their reconstructions. At the same time, the mode collapse issue is avoided and the training procedure is stabilized. The gated transformer also facilitates generating new styles through weighted connections between the transformer branches. Our discriminative network architecture has two components: the first to distinguish synthesized images from genuine images, and the other to identify the specific styles of these images. Experiments demonstrate that our adversarial gated networks successfully achieve multi-collection style transfer with a quality that is better or at least comparable to existing methods.

The remainder of this paper is organized as follows. In Section 2, we summarize related work. The proposed method is detailed in Section 3. The results of experiments using the proposed method and comparisons with existing methods are reported in Section 4. We conclude in Section 5.

## II. RELATED WORK

In this section, we introduce related style transfer works. We classify style transfer methods into four categories: texture synthesis-based methods, optimization-based methods, feedforward network-based methods, and adversarial network-based methods.

### A. Traditional Texture Transfer Method

Style transfer is an extension of texture transfer, the goal of the latter being to render an object with a texture taken from a different object [1], [2], [3], [4]. Most previous texture transfer algorithms rely on texture synthesis methods and low-level image features to preserve target image structure. Texture synthesis is the process of algorithmically constructing an unlimited number of images from a texture sample. The generated images are perceived by humans to be of the same texture but not exactly like the original images. A large range of powerful parametric and non-parametric algorithms exist to synthesize photo-realistic natural texture [18], [19], [20]. Based on texture synthesis, [21] and [22] used segmentation and patch matching to preserve information content. However, the texture transfer methods use only low-level target image features to inform texture transfer and take a long time to migrate a style from one image to another.

### B. Optimization-based Methods

The success of deep CNNs for image classification [23], [24] prompted many scientists and engineers to visualize features from a CNN [25]. DeepDream [24] was initially invented to help visualize what a deep neural network sees when given an image. Later, the algorithm became a technique to generate artworks in new psychedelic and abstract forms. Based on image representations derived from pre-trained CNNs, Gatys *et al.* [5] introduced a neural style transfer algorithm to separate and recombine image content and style. This approach has since been improved in various follow-up papers. Li *et al.* [26] studied patch-based style transfer by combining generative Markov random field (MRF) models and the pre-trained CNNs. Selim *et al.* [27] extended this idea to head portrait painting transfer by imposing novel spatial constraints to avoid facial deformations. Luan *et al.* [12] studied photorealistic style transfer by assuming the input to output transformation was locally affine in color space. Optimization-based methods can produce high quality results but they are computationally expensive, since each optimization step requires a forward and backward pass through the pre-trained network.

### C. Feedforward Networks-based Methods

Feedforward network-based methods accelerates the optimization procedure, which first iteratively optimizes a generative model and produces the styled image through a single forward pass. Johnson *et al.* [6] and Ulyanov *et al.* [7] trained a feedforward network to quickly produce similar outputs. Based on [7], Ulyanov *et al.* [11] then proposed to maximize quality and diversity by replacing the batch normalization module with instance normalization. After that, several works

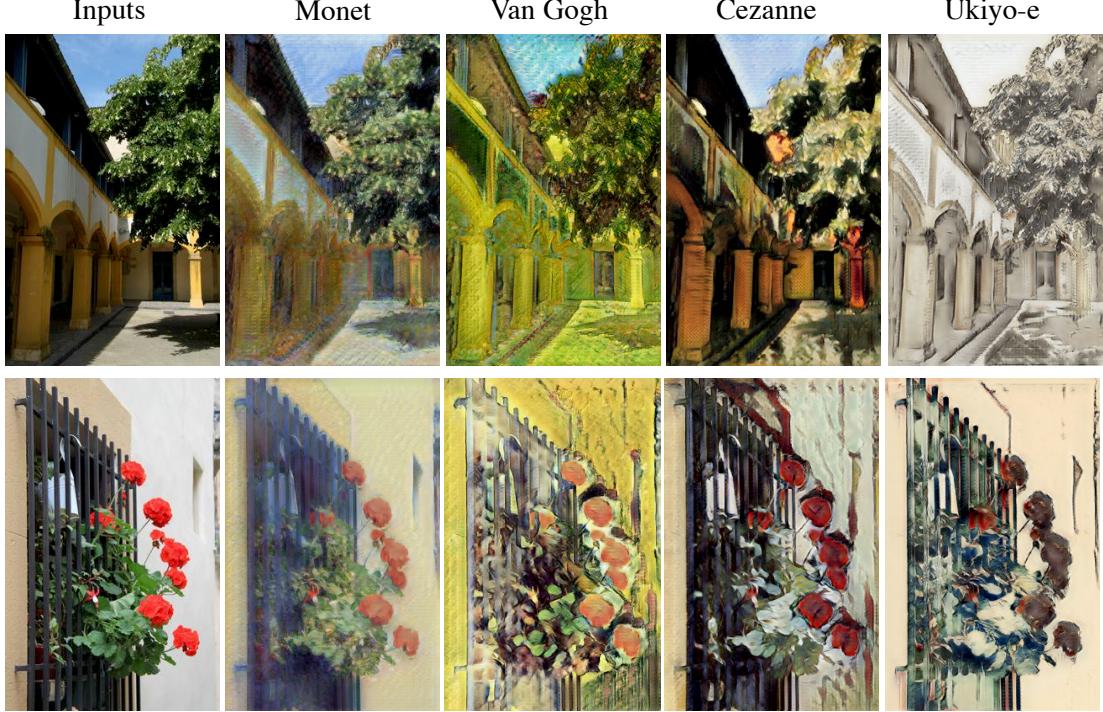


Fig. 1. Gated-GAN for multi-collection style transfer. The images are produced from a single model with a shared encoder and decoder are shared. Styles are controlled by switching different gated-transformer module. From left to right: original images, transferred images in Monet style, transferred images in Van Gogh's style, transferred images in Cezanne's style, transferred images in Ukiyoe-e's style.

explored multi-style transfer in a single network. Dumoulin *et al.* [28] proposed conditional instance normalization, which specialized scaling and shifting parameters after normalization to each specific texture and allowed the style transfer network to learn multiple styles. Huang *et al.* [29] introduced an adaptive instance normalization (AdaIN) layer that adjusted the mean and variance of the content input to match those of the style input. [8] introduced StyleBank, which was composed of multiple convolutional filter banks integrated in an auto-encoder, with each filter bank an explicit representation for style transfer. [9] took a noise vector and a selection unit as input to generate diverse image styles. Although adopting different methods to achieve multi-style transfer, they all explicitly extracted style presentations from style images based on the Gram matrix [5]. Gram matrix based methods could do collection style transfer if they use several images as style. Though those methods are designed to transfer the style of a single image, they could also transfer the style of several images by averaging their Gram matrix statistics of pretrained deep features. On the other hand, our methods learns to output samples in the distribution of the style of a collection. [30] achieved universal style transfer, by applying the style characteristics from a style image to content images in a style-agnostic manner. By whitening and coloring transformation, the feature covariance of content images could exhibit the same style statistical characteristics as the style images. In contrast, we are interested in the multi-collection style transfer problem. In contrast, we are interested in the multi-collection style transfer problem. A single image is difficult to comprehensively represent the style of an artist, and thus we study

multi-collection style transfer to abstract the style of an artist from a collection of images.

#### D. Adversarial Network-based Methods

GANs [13] represent a generative method using two networks, one as a discriminator and the other as a generator, to iteratively improve the model by a minimax game. Chuan *et al.* [31] proposed Markovian GANs for texture synthesis and style transfer, addressing the efficiency issue inherent in MRF-CNN-based style transfer [26]. Spatial GAN (SGAN) [32] successfully achieved data-driven texture synthesis based on GANs. PSGAN [33] improved Spatial GAN to learn periodical textures by extending the structure of the input noise distribution.

By adopting adversarial loss, many works have generated realistic images for conditional image generation, e.g., frame prediction [34], image super-resolution [35] and image-to-image translation [36]. However, these approaches often require paired images as input, which are expensive and hard to obtain in practice. Several studies have been conducted investigating domain transfer in the absence of paired images. [15], [16], [14] independently reported the similar idea of cycle-consistent loss to transform the image from the source domain to the target domain and then back to the original image. Taigman *et al.* [37] proposed Domain Transfer Network, which employed a compound loss function, including an adversarial loss and constancy loss, to transfer a sample in one domain to an analog sample in another domain.

In contrast, some works have generated different image types from noise in a single generative network. One strategy

was to supply both the generator and discriminator with class labels to produce class-conditional samples [38]. Another was to modify the discriminator to contain an auxiliary decoder network to output the class label for the training data [39], [40] or a subset of the latent variables from which the samples were generated [41]. AC-GAN [10] added auxiliary multi-class category loss to supervise the discriminator, which was used to generate multiple object types. Our work is different in that it focuses on exploring migrating different styles to content images.

### III. PROPOSED ALGORITHM

We first consider the collection style transfer problem. We have two sets of unpaired training samples: one set of input images  $\{x_i\}_{i=1}^N \in X$  and the target set of collections for artist or genre  $\{y_i\}_{i=1}^M \in Y$ . We aim to train a generative network that generates images  $G(x)$  in the style of a target artist or genre, and simultaneously we train a discriminative network  $D$  to distinguish the transferred images  $G(x)$  from the real style image  $y$ . The generative network implicitly learns the target style from *adversarial loss*, aiming to fool the discriminator. The whole framework has three modules: an encoder, a gated-transformer and a decoder. The encoder consists of a series of convolutional layers that transform input image into feature space  $Enc(x)$ . After the encoder, a series of residual networks [42] become the transformer:  $T(\cdot)$ . The input of residual layer in gated function  $T$  is the feature maps from the last layer of encoder module  $Enc(x)$ . The output of the gated function is the activations  $T(Enc(x))$ . Then, a series of fractionally-strided convolutional networks decode the transformed feature into output images  $G(x) = Dec(T(Enc(x)))$ . To stabilize training, we introduce the *auto-encoder reconstruction loss*. We introduce the gated transformer module to integrate multiple styles within a single generated network. The network architecture is shown in Figure 2, and the overall architecture is called the adversarial gated network (Gated-GAN).

#### A. Adversarial Network for Style Transfer

To learn a style from the target domain  $Y$ , we apply adversarial loss [13], which simultaneously trains  $G$  and  $D$  as the two-player minimax game with loss function  $L(G, D)$ . The generator  $G$  tries to generate an image  $G(x)$  that looks similar in style to target domain  $Y$ , while the discriminator  $D$  aims to distinguish between them. Specifically, we train  $D$  to maximize the probability of assigning the correct label to target image  $y$  and transferred image  $G(x)$ , meanwhile training  $G$  to minimize the probability of the discriminator assigning the correct label to transferred image  $G(x)$ . The original generative adversarial value function is expressed as follows:

$$\min_G \max_D V(G, D) = \mathbb{E}_{y \in Y} [\log D(y)] + \mathbb{E}_{x \in X} [\log(1 - D(G(x)))] \quad (1)$$

We employ the least squares loss (LSGAN) as explored in [43], which provides a smooth and non-saturating gradient

in the discriminator  $D$ . The adversarial loss  $\mathcal{L}_{GAN}(G, D)$  becomes:

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{y \in Y} [(D(y) - 1)^2] + \mathbb{E}_{x \in X} [D(G(x))^2]. \quad (2)$$

#### B. Auto-encoder Reconstruction Loss for Training Stabilization

The original GAN framework is known to be unstable, as it must train two neural networks with competing goals. [14] pointed out that one reason for instability is that there exist non-unique solutions when the generator learns the mapping function. Due to unpaired training samples, the same set of input images can be mapped to any random permutation of images in the target domain. To reduce the space of possible mapping functions, we introduce the *auto-encoder reconstruction loss*. In our model, the auto-encoder is obtained by directly connecting the encoder and decoder modules. That is, the network is encouraged to produce output  $Dec(Enc(x))$  identical to input image  $x$  after learning the representation (encoding:  $Enc(x)$ ) for the input data. We define the L1 loss between the reconstructed output and input as the *auto-encoder reconstruction loss*:

$$\mathcal{L}_R = \mathbb{E}_{x \in X} [| | | Dec(Enc(x)) - x | |_1]. \quad (3)$$

Mode collapse is a common problem in vanilla GAN [44], where all input images might be mapped to the same output image, and the optimization fails to make progress. In collection style transfer, if the networks trained with adversarial loss alone have sufficient capacity, content images would be mapped to an arbitrary output as long as it matches the target style. The proposed encoder-decoder subnetwork aims to reconstruct input images, so that structures of the output are expected to be consistent with the input image, which guarantees diversity of the output along with different inputs.

#### C. Adversarial Gated Network for Multi-Collection Style Transfer

1) *Gated Generated Network*: In multi-collection style transfer, we have a set of input images  $\{x_i\}_{i=1}^N \in X$  and collections of paintings  $Y = \{Y_1, Y_2, \dots, Y_K\}$ , where  $K$  denotes number of collections. In each collection, we have  $M_c$  numbers of images  $\{y_i\}_{i=1}^{M_c} \in Y_c$ , where  $c$  indicates the index of collection. The proposed gated generative network aims to output images  $G(x, c)$  by assigning specific style  $c$ . Specifically, the gated-transformer (red blocks in Figure 2) transforms the input from encoded space into different styles by switching trigger to different branches:

$$G(x, c) = Dec(T(Enc(x), c)). \quad (4)$$

In each branch, we employ the residual network as the transfer module. The encoder and decoder are shared by different styles, so the network only has to save the extra transformer module parameters for each style.

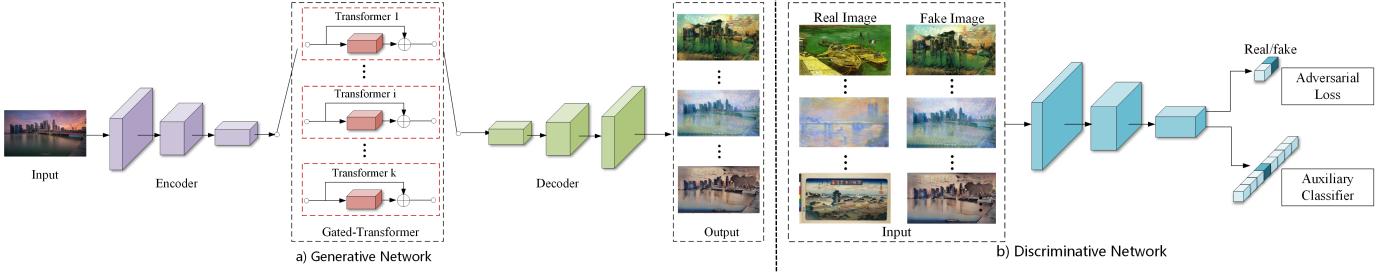


Fig. 2. Architecture of the proposed adversarial gated networks: a generative network and a discriminative network. The generative network consists of three modules: an encoder, a gated transformer, and a decoder. Images are generated to different styles through branches in the gated transformer module. The discriminative network uses adversarial loss to distinguish between stylized and real images. An auxiliary classifier supervises the discriminative network to classify the style categories.

2) *Auxiliary Classifier for Multiple Styles*: If we only use the adversarial loss, the model tends to confuse and mix multiple styles together. Therefore, we need a supervision to separate categories of styles. One solution is to adopt LabelGAN [40]. [40] generalized binary discriminator to multi-class case with its associated class label  $c \in \{1, \dots, K\}$ , and the  $(K+1)$ -th label corresponds to the generated samples. The objective functions are defined as:

$$\mathcal{L}_G^{lab} = \mathbb{E}_{x \in X} [H([1, 0], [D_r(G(x)), D_{K+1}(G(x))])], \quad (5)$$

$$\mathcal{L}_D^{lab} = \mathbb{E}_{(y,c) \in Y} [H(v(c), D(y))] + \mathbb{E}_{x \in X} [H(v(K+1), D(G(x)))] \quad (6)$$

where denotes the probability of the sample  $x$  to have the  $i$ -th style.  $D(x) = [D_1(x), D_2(x), \dots, D_{K+1}(x)]$  and  $v(c) = [v_1(c), v_2(c), \dots, v_{K+1}(c)]$  with  $v_i(c) = 0$  if  $i \neq c$  and  $v_i(c) = 1$  if  $i = c$ .  $H$  is the cross-entropy, defined as  $H(p, q) = -\sum_i p_i \log q_i$ . In LabelGAN, the generator gets its gradients from the  $K$  specific real class logits in discriminator and tends to refine each sample towards being one of the classes. However, LabelGAN actually suffers from the overlaid-gradient problem [45]: all real class logits are encouraged at the same time. Though it tends to make each sample be one of these classes during the training, the gradient of each sample is a weighted averaging over multiple label predictors.

In our method, an auxiliary classifier (denoted as  $C$ ) is added in the consideration of leveraging the side information directly:

$$\mathcal{L}_G^{Gated} = \lambda_{CLS} \mathbb{E}_{x \in X} [H(u(c), C(G(x, c)))] + \mathcal{L}_{GAN} \quad (7)$$

where  $u(\cdot)$  is the vectorizing operator that is similar to  $v(\cdot)$  but defined with  $K$  classes, and  $C(G(x, c))$  is the probability distribution over  $K$  real classes given by the auxiliary classifier.  $\mathcal{L}_{GAN}$  indicates the adversarial loss (in Equation 2) that encourages to generate realistic images. In the first term of Equation 7, we optimize entropy to make each sample have a high confidence of being one of the classes, so that the overlaid-gradient problem can be overcome. The loss can be written in the form of log-likelihood:

$$\min_C \mathcal{L}_{CLS}(C) = -\mathbb{E}_{(y,c) \in Y} \{\log C(Style = c|y)\}. \quad (8)$$

The classifier  $C$  is encouraged to correctly predict the log-likelihood of the correct class given real images. Meanwhile,

the generator aims to generate images that can be correctly recognized by classifier:

$$\min_G \mathcal{L}_{CLS}(G) = -\mathbb{E}_{x \in X} \{\log C(Style = c|G(x, c))\}. \quad (9)$$

In practice, the classifier shares low-level convolutional layers with the discriminator, but they have exclusive fully connected layers to output the conditional distribution.

#### IV. IMPLEMENTATION

1) *Network Configuration*: Our generative network architecture contains two stride-2 convolutions (encoder), one gated residual blocks (gated-transfer), five residual blocks, and two fractionally-convolutions with  $\frac{1}{2}$  stride (decoder). Instance normalization [40] is used after the convolutional layers. Details are provided in Table I.

For the discriminators and classifiers, we adapt the Markovian Patch-GAN architecture [31], [36], [14], [16]. Instead of operating over the full images, the discriminators and classifiers distinguish overlapping patches, sampling from the real and generated images. By doing so, the discriminators and classifiers focus on local high-frequency features like texture and style and ignore the global image structure. The patch size is set to  $70 \times 70$ . In addition, PatchGAN has fewer parameters and can be applied to any size of input.

2) *Training Strategy*: To smooth the generated image  $G(x, c)$ , we make use of the *total variation loss* [6], [46], [47], denoted by  $\mathcal{L}_{TV}$ :

$$\mathcal{L}_{TV} = \sum_{i,j} \left[ (G(x)_{i,j+1} - G(x)_{i,j})^2 + (G(x)_{i+1,j} - G(x)_{i,j})^2 \right]^{\frac{1}{2}} \quad (10)$$

where  $i \in (0, \dots, H-1)$  and  $j \in (0, \dots, W-1)$  and  $G(x)$  is the generated image whose dimension is  $H \times W$ . The full objective of the generator is minimizing the loss function:

$$\mathcal{L}(G) = \mathcal{L}_{GAN} + \lambda_{CLS} \mathcal{L}_{CLS} + \lambda_{TV} \mathcal{L}_{TV} \quad (11)$$

where  $\lambda_{CLS}$  and  $\lambda_{TV}$  are parameters that control relative importance of their corresponding loss functions. Alternatively, we train an auto-encoder by minimizing the weighted reconstruction loss in Equation 3:  $\lambda_R \mathcal{L}_R$ . The discriminator maximizes the prediction of real images and generated images  $\mathcal{L}(D) = \mathcal{L}_{GAN}$ , while the classifier in Equation 8 maximizes the prediction of collections from different artists or genres.

TABLE I  
GENRATIVE NETWORK OF GATED-GAN

	Operation	Kernel size	Stride	Feature maps	Normalization	Nonlinearity
Encoder	Convolution	7	1	32	Instance Normalization	ReLU
	Convolution	3	2	64	Instance Normalization	ReLU
	Convolution	3	2	128	Instance Normalization	ReLU
Gated-transformer	Residual block			128	Instance Normalization	ReLU
Decoder	Residual block			128	Instance Normalization	ReLU
	Residual block			128	Instance Normalization	ReLU
	Residual block			128	Instance Normalization	ReLU
	Residual block			128	Instance Normalization	ReLU
	Residual block			128	Instance Normalization	ReLU
	Fractional-convolution	3	1/2	64	Instance Normalization	ReLU
	Fractional-convolution	3	1/2	32	Instance Normalization	ReLU
	Convolution	7	1	3	-	tanh

For all experiments, we set  $\lambda_{CLS} = 1$ ,  $\lambda_R = 10$ , and  $\lambda_{TV} = 10^{-6}$ . The networks are trained with a learning rate of 0.00002, using the Adam solver [48] with batch size of 1.

The input image is  $128 \times 128$ . The training samples are first scaled to  $143 \times 143$ , and then randomly flipped and cropped to  $128 \times 128$ . We train our model with input size of  $128 \times 128$  based on two reasons. First, randomly cropping raw input could augment the number of training set. Secondly, a relatively smaller size of image decreases the computational cost, so that speeds up training procedure. In test phase, We test images with their original resolution to receive a clearer exhibition in the paper.

To stabilize training, we update the discriminative networks using a history of transferred images rather than the ones produced by the latest generative network [49]. Specifically, we maintain an image buffer that stores 50 previously generated images. At each iteration of discriminator training, we compute the discriminator loss function by sampling images from the buffer. The training process is shown in Algorithm 1.  $\theta_{Enc}$  denotes the parameter of encoder module and  $\theta_{Dec}$  denotes the parameters of decoder module. In practice,  $K_g$  and  $K_d$  are set to 1.

## V. EXPERIMENTS

In this section, we evaluate the effectiveness, stability, and functionality of the proposed model. We first introduce a quantitative assessment of image quality. Then, we set up a texture synthesis experiment and visualize the filters in the gated transformer branches. Lastly, we train the model for multiple style transfer and compare results with state-of-the-art algorithms.

### A. Assessment of Image Quality

We used FID score [50] to quantitatively evaluate the quality of results. FID score measures the distance between the generated distribution and the real distribution. To this end, the generated samples are first embedded into a feature space given by (a specific layer) of Inception Net. Then, taking the embedding layer as a continuous multi-variate Gaussian, the mean and covariance are estimated for both the generated data and the real data. The Frchet distance between these two

---

### Algorithm 1 Adversarial training of gated network $G$ .

---

**Require:** The set of training sample  $\{x_i\}_{i=1}^N \in X$ , The set of style images with category  $\{y_i, c_i\} \in Y$ , number of discriminator network updates per step  $K_d$ , number of generative network updates per step  $K_g$ .

**Ensure:** Gated generative newtworks:

```


$$G = Dec(T(Enc(\cdot), \cdot)).$$

1: for number of training iterations do
2:   for  $K_d$  steps do
3:     Sample minibatch of style images  $(y_i, c_i)$  and
       training images  $x_i$ .
4:     Generate stylized image  $G(x_i, c_i)$  in Equation 4.
5:     Update discriminator  $D$  and classifier  $C$ 
      
$$\Delta_{\theta_D} \leftarrow \nabla_{\theta_D} \mathcal{L}_{GAN}, \Delta_{\theta_C} \leftarrow \theta_D \nabla_{\theta_C} \mathcal{L}_{CLS}.$$

6:   end for
7:   for  $K_g$  steps do
8:     Sample training images  $x_i$ 
9:     Update generator  $G$  :
      
$$\Delta_{\theta_G} \leftarrow \nabla_{\theta_G} (\mathcal{L}_{GAN} + \lambda_{CLS} \mathcal{L}_{CLS} + \lambda_C \mathcal{L}_C + \lambda_{TV} \mathcal{L}_{TV}).$$

10:  end for
11:  Update encoder and decoder module  $\theta_{Enc}, \theta_{Dec}$ :
      
$$\Delta_{\theta_{Enc}, \theta_{Dec}} \leftarrow \nabla_{\theta_{Enc}, \theta_{Dec}} (\lambda_R \mathcal{L}_R).$$

12: end for

```

---

Gaussians is then used to quantify the quality of the samples, i.e.,

$$FID(x, g) = \|\mu_x - \mu_g\|_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}) \quad (12)$$

where  $(\mu_x, \Sigma_x)$  and  $(\mu_g, \Sigma_g)$  are the mean and covariance of sample embeddings from the real data distribution and generative model distribution, respectively. In our experiment, we use paintings of artists as samples of real distribution and stylized images as samples of generated distribution. That is to say, we compute the FID between generated images and authentic work of painting.

### B. Texture synthesis

To explicitly understand the gated-transformer module in the proposed Gated-GAN, we design an experiment to explore what the gated-transformer learns. We use our Gated-GAN to achieve synthesize texture, and visualize the gated-transformer filters. For each style, the training set is a textured image.



Fig. 3. Four cases of texture synthesis using Gated-GAN. For each case, the first column shows examples of texture, and the other three are synthesized results given different samples of Gaussian noise as inputs.

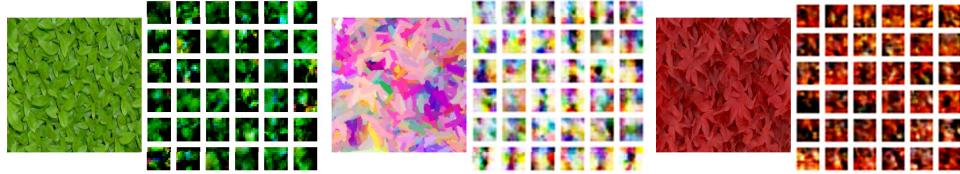


Fig. 4. Visualization of learned features in the gated transformer of the generative networks. In each case, the left shows synthesized images and the right shows the corresponding features.

The training samples are first scaled to  $143 \times 143$ , and then randomly flipped and cropped to  $128 \times 128$ . The generative network input is Gaussian noise. After adversarial training, the generative network outputs realistic textured images (see Figure 3).

To explore style representations learned from the gated-transformer, we visualize the transformer filters in Figure 4. The features are decoded by  $3 \times 3 \times 128$  tensors, where only one of the 128 channels is activated by Gaussian noise. They passed through different gated transformer filters but the same decoder. Since the output of decoder contains three channels (RGB channels), we are able to observe the colour of output decoded from the learned feature. This reveals that the transformer module learns style representations, e.g., color, stroke, etc. Another interpretation is that the transformer module learns the bases or elements of styles. Generated images can be viewed as linear combinations of these bases, with coefficients learned from the encoder module.

### C. Style Transfer

In this subsection, we present our results for generating multiple styles of artists or genres using a single network. Then, we compare our results with state-of-the-art image style transfer and collection style transfer algorithms. The model is trained to generate images in style of Monet, Van Gogh, Cezanne, and Ukiyo-e, whose datasets are from [14]. Each contains 1073, 400, 526, and 563 paintings, respectively.

1) *Multi-Collection Style Transfer*: Collection style transfer mimics the style of artists or genres with respect to their features, e.g., stroke, impasto, perspective frame usage, etc. Figure 5 shows the results of collection style transfer using our method. Original images are presented on the left, and the generated images are on the right. For comparison, Monets



Fig. 5. Collection style transfer on Photo → Monet. From left to right: input photos, Monet's paintings picked from a similar landscape theme, and our stylized images. The photo is transferred adaptively based on different themes.

paintings depicting similar scenes are shown in the middle. It can be seen that the styles of the generated images and their corresponding paintings are similar. Although the themes and colors of the two generated images are different, they still appear similar to Monet's authentic pieces. Our method can clearly mimic the style of the artist for different scenes. Figure 6 shows the results of applying the trained network on evaluation images for Monet, Van Gogh, Cezanne, and Ukiyo-e styles.

2) *Comparison with Image Style Transfer*: The image style transfer algorithm [5] focuses on producing images that combine the content of an arbitrary photograph and style of one or many well-known artworks. This is achieved by minimizing

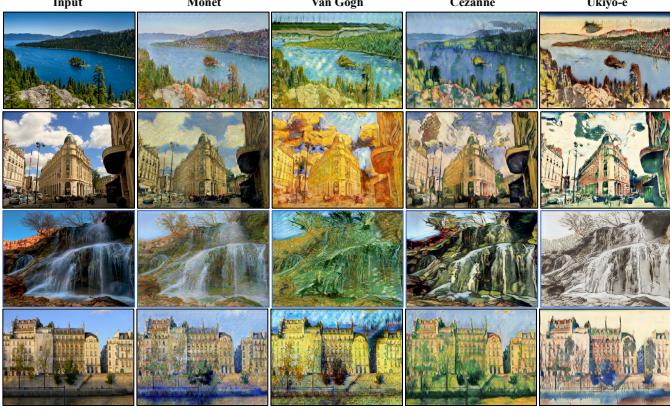


Fig. 6. A four-style transfer network is trained to capture the styles of Monet, Van Gogh, Cezanne, and Ukiyo-e.

the mean-squared distance between the entries of the Gram matrix from the style image and the Gram matrix of the image to be generated. We note some recent works on multi-style transfer [28], [8], [19], but these are all based on neural style transfer [5]. Thus, we compare our results with [5].

For each content image, we use two representative artworks as the reference style images. To generate images in the style of the entire collection, the target style representation is computed by the average Gram matrix of the target domain. To compare this with our method, we use the collections of artists artworks or a genre and compute the average style as the target.

Figure 7 reports the difference of methods. We can see that Gatys et al. [5] requires manually picking target style images that closely match the desired output. If the entire collection is used as target images, the transferred style is the average style of the collections. In contrast, our algorithm outputs diverse and reasonable images, each of which can be viewed as a sample from the distribution of the artist’s style.

3) *Comparison with Universal Style Transfer:* [30] aims to apply the style characteristics from a style image to content images in a style-agnostic manner. By whitening and coloring transformation, the feature covariance of content images could exhibit the same style statistical characteristics as the style images without requiring any style-specific training.

We compare images generated from the proposed algorithm and those from [30]. The results are shown in Figure 8. Given a picture with bushes and flowers (see Figure 8 (a)), our method outputs what Monet might record this scenery (see Figure 8 (d)), in which the style of painting bushes and flowers is similar to Monets painting of “Flowers at Vetheuil”. What if the content image is a cityscape? Our method outputs images with foggy strokes (see Figure 8 (h)), since Monet produced a lot of cityscapes with fog in London (e.g. “Charing Cross Bridge”). On the other hand, [30] transfers images by following a particular style image. Taking “Flowers at Vetheuil” as the style image, Figure 8 (g) produced by [30] well inherits the style of Monets “Flowers at Vetheuil” with green and red spot. However, Monet might not paint a cityscape with green and red spot as painting flowers.

TABLE II  
QUANTITATIVE EVALUATION ON COLLECTION STYLE TRANSFER IN TERMS OF FID TO MEASURE THE PERFORMANCE. LOWER SCORE INDICATES BETTER QUALITY.

Style	Content Images	CycleGAN [14]	Ours
Monet	86.50	64.14	55.13
Cezanne	186.73	106.96	107.27
Van Gogh	173.01	107.03	109.59
Ukiyoe	195.25	103.36	115.96
MEAN	160.37	95.37	96.99

In summary, our task focuses on what the artists or genres might paint given content images, while the task of [30] is to apply style characteristics from a particular style image to any content images. Both [30] and our method output interesting results, and could be used in different scenarios.

4) *Comparison with Collection Style Transfer:* CycleGAN [14] previously showed impressive results on collection style transfer, so in this section we compare our results with CycleGAN. The generative network of baseline CycleGAN is composed of three stride-2 convolutional layers, 6 residual blocks, two fractional-convolutional layers and one last convolutional layer, which shares the same structure with our method in our experiment. Figure 9 demonstrates multi-collection style transfer by our method, which shows that the proposed model produces comparable results to CycleGAN.

Quantitative results are shown in Table II, though the quality of images generated from the proposed algorithm exhibits similar performance as those of CycleGAN, it is instructive to note that our four styles are produced from a single network. In the second column of Table II, we compute the score of the corresponding content images of stylized images. We find that the stylized images achieve better performance than original content images. It demonstrates that the stylized images are more similar to the real authentic work of artists, which is consistent with our intuitive expectation.

Finally, we compare model size with CycleGAN [14]. The generative network is composed of several convolutional layers and residual blocks with the same architecture as Gated-GAN when the transformer module number is set to one. The parameters of the two models are the same. Given another  $N$  styles, CycleGAN must train another  $N$  models. A whole generative network must be included for a new style. For Gated-GAN, the transformation operator is encoded in the gated transformer, which only has one residual block. A new style will thus only require a new transformer part in the generative network. As a result, the proposed method saves storage space as the style number increases. In Figure 10, we compare the numbers of parameters with those of CycleGAN. Both models are trained for  $128 \times 128$  training images.

5) *Comparison with Conditional GAN:* Conditional GAN [10], [38] model is a widely used method to generate class-conditional image. When the conditional GAN is applied in multi style transfer, a stylized image  $G(c, x)$  is generated from a content image  $x$  and a style class label  $c$ . We compare conditional GAN in experiments. Class label is represented by a one-hot vector with  $k$  bit where each represents a style type.  $k$  noise vectors of the same dimension as the content

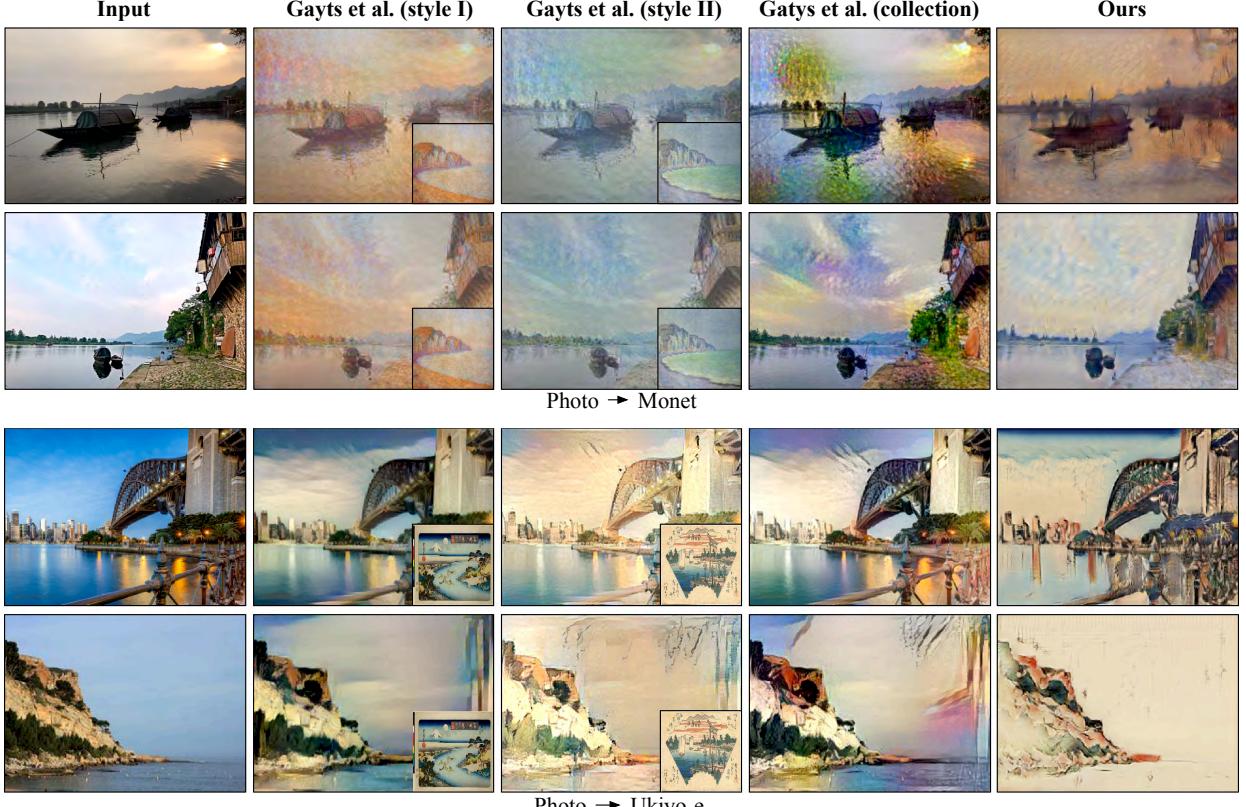


Fig. 7. Comparison of our methods with image style transfer [5] on photo → Monet and photo → Ukiyo-e. From left to right: input photos, Gatys *et al.*'s results using different target style images, Gatys *et al.*'s results using the entire collection of artist and genre, our results for collection style transfer.

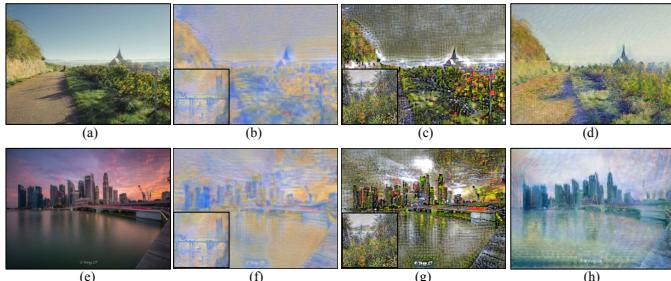


Fig. 8. Comparison of our methods with universal style transfer [30] on photo → Monet. From left to right: input images, results of [30] with the style image: Monet *Charing Cross Bridge*, results of [30] with the style image: Monet *Flowers at Vetheuil*, and our results of Monet's collection style transfer.

image are randomly sampled from a uniform distribution. The input of generative network is obtained by concatenating the content image with the outer product of these noise vectors and the class label.

As we can see in Figure 11 (b), the conditional GAN fails to output meaningful results. This is because in collection style transfer, conditional GAN lacks of paired input-output examples. To stabilize the training of conditional GAN, we adopt cycle-consistent loss [14]. From the results of conditional GAN with cycle-consistent loss in Figure 11 (c), we can see that the results of different styles tend to be much similar, and only colors are changed at first sight. In contrast, our results (see Figure 11 (d)) are more diverse in different styles in terms of strokes and textures.

TABLE III  
QUANTITATIVE EVALUATION OF PARAMETER  $\lambda_{CLS} = \{0, 0.1, 1, 5, 10\}$  IN TERMS OF FID SCORE.

Style	$\lambda_{CLS} = 0$	$\lambda_{CLS} = 0.1$	$\lambda_{CLS} = 1$	$\lambda_{CLS} = 5$	$\lambda_{CLS} = 10$
Monet	204.82	63.35	55.13	62.66	61.48
Cezanne	234.02	136.35	107.27	127.77	143.39
Van Gogh	217.10	112.61	109.59	126.56	138.66
Ukiyoe	206.67	138.13	115.96	132.72	140.53
MEAN	215.65	112.61	96.99	112.42	121.02

#### D. Analysis of Loss Function

1) *Influence of Parameters in Loss Function:* In our model, we proposed an auxiliary classifier loss and an auto-encoder reconstruction loss, which are balanced by parameters  $\lambda_{CLS}$  and  $\lambda_R$  respectively. Now we analyze the influence of parameters. To explore the influence of parameters, we do experiments by considering  $\lambda_{CLS} = \{0, 0.1, 1, 5, 10\}$  and  $\lambda_R = \{1, 5, 10, 20\}$ .

Figure 13 and Table IV demonstrate the qualitative and quantitative comparisons of the influence of parameter  $\lambda_{CLS}$ . We can see the classifier loss provides a supervision of styles. Without classifier loss ( $\lambda_{CLS} = 0$ ), our model will only transfer into one style. If we set too large ( $\lambda_{CLS} = 10$ ), the model would produce images with some artifacts. The underlying reason is that larger  $\lambda_{CLS}$  suppresses the function of discriminative network so that the output becomes less realistic. As a result, we set  $\lambda_{CLS} = 1$  in our model.

Figure 13 and Table IV reveal the qualitative and quantitative comparisons of the influence of parameter  $\lambda_R$ . We can see that if we set  $\lambda_R$  too small ( $\lambda_R = 1$ ), the outputs tend

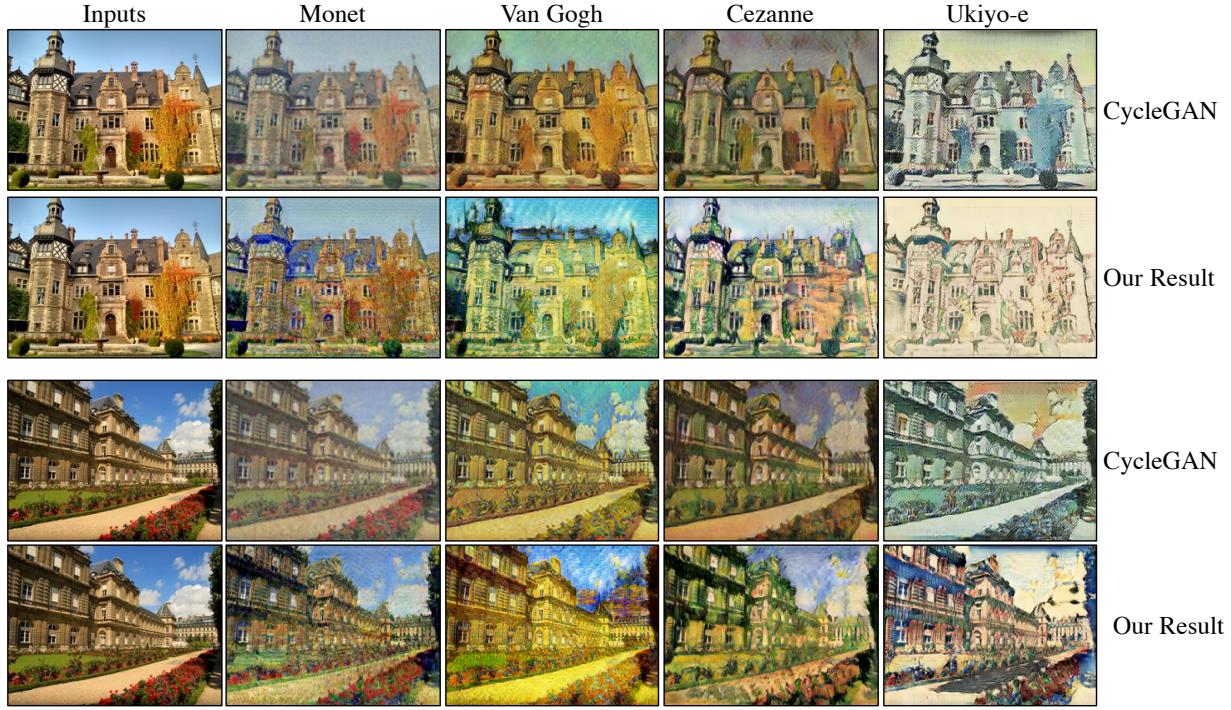


Fig. 9. Comparison with CycleGAN [14]. From left to right: original images, stylized images in Monet’s style, stylized images in Van Gogh’s style, stylized images in Cezanne’s style, stylized images in Ukiyo-e style. In each case, the first row shows the results produced by CycleGAN, and the second row shows our results.

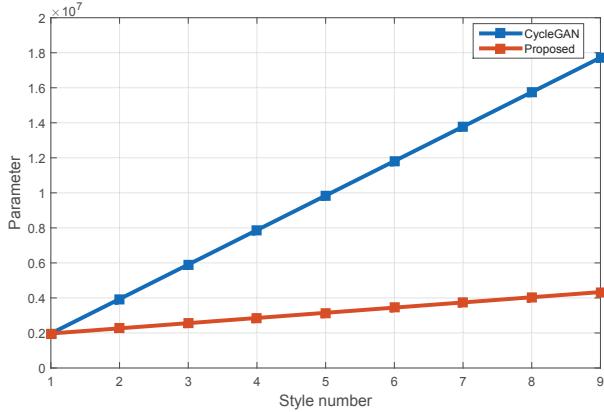


Fig. 10. Model size. We compare the number of parameters between our model and CycleGAN [14]. The x-axis indicates style number and the y-axis indicates the model size.

to be blurry and meaningless. It is because  $\lambda_R$  improve the stability of training procedure. When  $\lambda_R = \{5, 10, 20\}$  the visual qualities are similar while the FID score shows achieves a slightly better quantitative performance. It demonstrates that our method is robust and easy to reproduce satisfying results. Since  $\lambda_R = 10$  achieves the best quality, we set  $\lambda_R = 10$  in our model.

2) *Analysis of Auto-encoder Reconstruction Loss:* We next justify our choice of L1-norm. Beyond L1-norm, L2-norm can also be used in Equation 3. In Table V, we find that there is no significant difference between results of L1 and L2 loss. In CycleGAN [14], L1-norm is used in cycle-consistent reconstruction loss. As CycleGAN is an important comparison

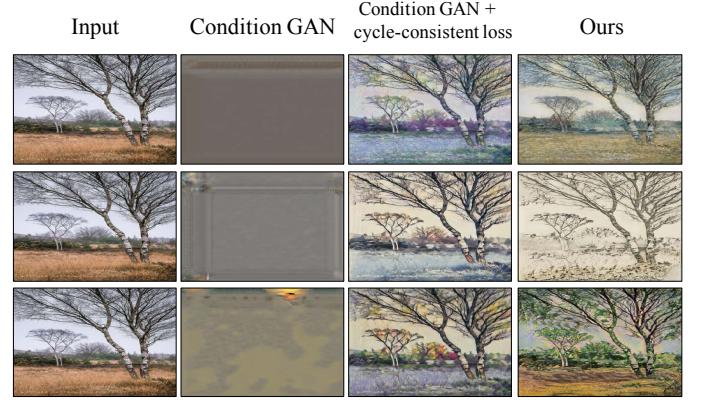


Fig. 11. Comparison of our methods with Condition GAN and its variant. From left to right: input, condition GAN and condition GAN + cycle-consistent loss. Each row indicates different styles, from top to bottom: Monet, Ukiyo-e, Cezanne.

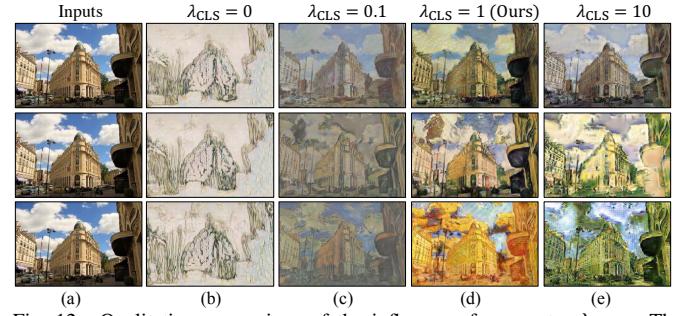


Fig. 12. Qualitative comparison of the influence of parameter  $\lambda_{CLS}$ . The first column shows the input images. The rest columns demonstrate results with  $\lambda_{CLS} = \{0, 0.1, 1, 10\}$ . Each row demonstrates images transferred by different styles. From top to bottom: Monet, Cezanne, Van Gogh.

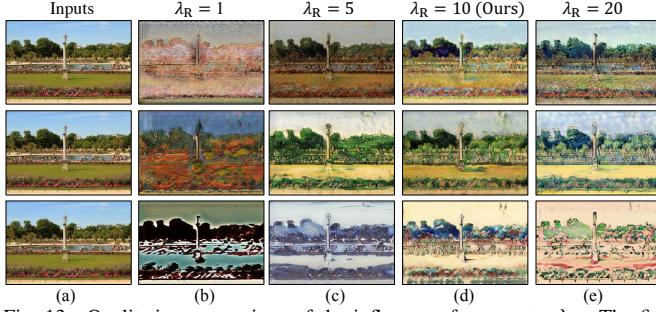


Fig. 13. Qualitative comparison of the influence of parameter  $\lambda_R$ . The first column shows the input images. The rest columns demonstrate results with  $\lambda_R = \{1, 5, 10, 20\}$ . Each row demonstrates images transferred by different styles. From top to bottom: Monet, Cezanne, Ukiyo-e.

TABLE IV

QUANTITATIVE EVALUATION OF PARAMETER  $\lambda_R = \{1, 5, 10, 20\}$  IN TERMS OF FID SCORE.

Style	$\lambda_R = 1$	$\lambda_R = 5$	$\lambda_R = 10$	$\lambda_R = 20$
Monet	180.30	121.07	55.13	115.09
Cezanne	165.27	148.67	107.27	140.84
Van Gogh	148.43	139.87	109.59	134.13
Ukiyoe	166.69	134.26	115.96	138.54
MEAN	165.17	135.97	96.99	132.15

algorithm in our paper, we adopt L1-norm in our auto-encoder reconstruction loss as well.

Lastly, we analyze the influence of the *auto-encoder reconstruction loss* in stabilizing the adversarial training procedure. We train a comparative model by ignoring the *auto-encoder reconstruction loss* in Equation 3. In Figure 14, the model without Equation 3 generates images with random texture and tend to be less diverse after training for several iterations. In contrast, the full proposed model generates satisfying results. Without the *auto-encoder reconstruction loss*, the network only aims to generate images to fool the discriminative network, which often leads to the well-known problem of mode collapse [44]. Our encoder-decoder subnetwork is encouraged to reconstruct input images, and thus semantic structure of the input is aligned with that of the output, which directly encourages diversity of output along with different inputs. As a result, the full proposed model outputs satisfying results.

#### E. Analysis of network architecture

We explore the influence of neural network structure. We setup variants of our model in Table VI. Variants of models have different configurations of gated-transformer module. The quantitative results in Table VII reveal that the performance

TABLE VI  
EXPERIMENT SETUP OF NETWORK ARCHITECTURE ANALYSIS

	Expt1	Expt2	Expt3
Encoder	3 × Convolution	3 × Convolution	3 × Convolution
Gated-transformer	1 × Residual block	1 × Convolution	2 × Residual block
Decoder	5 × Residual block 2 × Fractional-convolution 1 × Convolution	5 × Residual block 2 × Fractional-convolution 1 × Convolution	5 × Residual block 2 × Fractional-convolution 1 × Convolution

TABLE VII  
QUANTITATIVE EVALUATION ON DIFFERENT NETWORK STRUCTURE IN TERMS OF FID.

Style	Variants 1(Ours)	Variants 2	Variants 3
Monet	55.13	67.16	53.08
Cezanne	107.27	128.62	110.13
Van Gogh	109.59	199.71	109.07
Ukiyoe	115.96	195.87	100.32
MEAN	96.99	147.84	93.15

of variant 2 declines compared to that of the variant 1. From qualitative results in Figure 15, we observe that model of variant 2 cannot maintain content structure (see Figure 15 (c)). The underlying reason is that the residual block has a branch that skip the convolutional layer and directly connects between the encoder and decoder module. Since the encoder-decoder subnetwork learns the content information of input from reconstruction loss, residual blocks with skipping connection shuttle the encoded information to the decoder module, which helps our model to output results aligned with the structured of input images.

To analyze the influence of layer size of gated-transformer module, we set variant 3 whose gated-transformer consists of 2 residual blocks. In Table VII, we can see the model of variant 3 achieves a slightly better quantitative evaluation than variant 1. The reason is that with the number of residual blocks increasing, the expression capacity of network increases as well, which means the model could capture more details for each style. However, the performance rise of variant 3 is limited and the qualitative qualities are similar in Figure 15, which means one residual block of gated-transformer is sufficient in multi style transfer. As a result, we adopt variant 1 of architecture as our method.

#### F. Incremental Training

By sharing the same encoding/decoding subnets, our model is compatible to the new style. For a new style, our model enables to add the style by learning a new branch in the gated-transformer while holding the encoding-decoding subnets fixed. We first jointly train the encoder-decoder subnetwork and gated-transformer (three collection style: Cezanne, Ukiyoe and Van Gogh) with the strategy described in Algorithm 1. After that, for new style (Monet), we train a new branch of residual blocks in the gated-transformer.

Figure 16 shows several results of new style by incremental training. It obtains very comparable stylized results to the CycleGAN, which trains the whole network with the style. We also evaluate the quantitative performance of the new style in term of FID score. The new style by incremental training gets score of 57.27. Compared to 55.13 of our Gated-GAN and

TABLE V  
QUANTITATIVE EVALUATION ON L1-NORM AND L2 NORM IN TERMS OF FID SCORE.

Style	CycleGAN	Ours (L1-norm)	Ours (L2-norm)
Monet	64.14	55.13	56.09
Cezanne	106.96	107.27	101.54
Van Gogh	107.03	109.59	109.33
Ukiyoe	103.36	115.96	112.39
MEAN	95.37	96.99	94.84

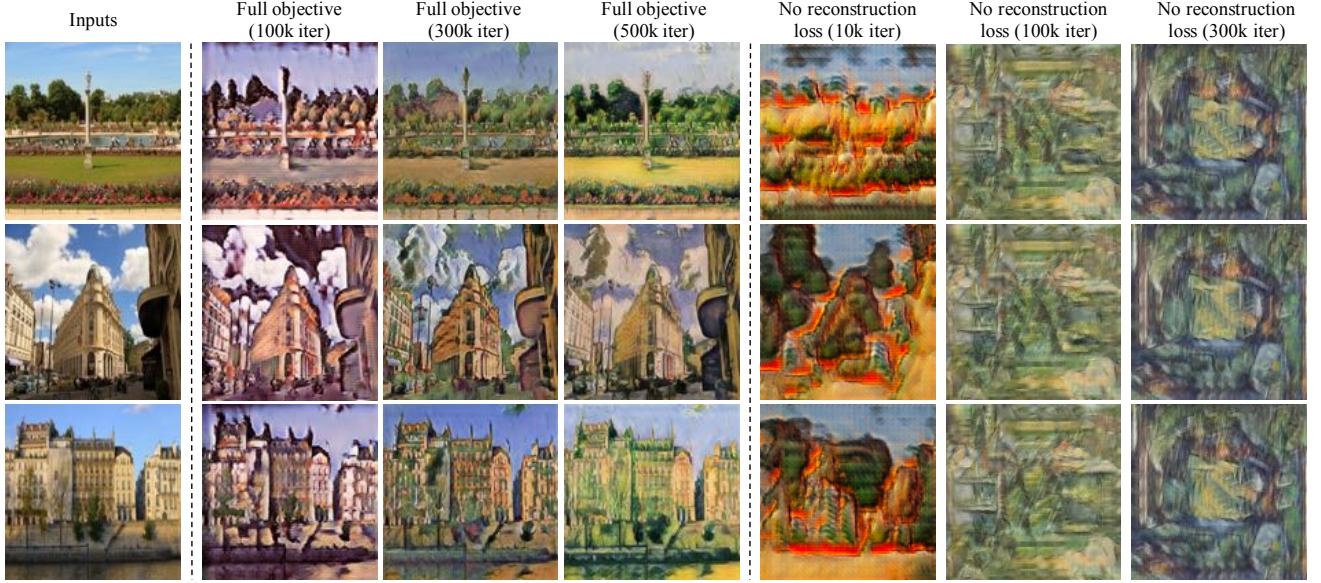


Fig. 14. Comparison with a variant of our method across different training iterations for mapping images to Cezanne's style. From left to right: original images, results after training for 10k, 100k, and 300k iterations with and without auto-encoder reconstruction loss.

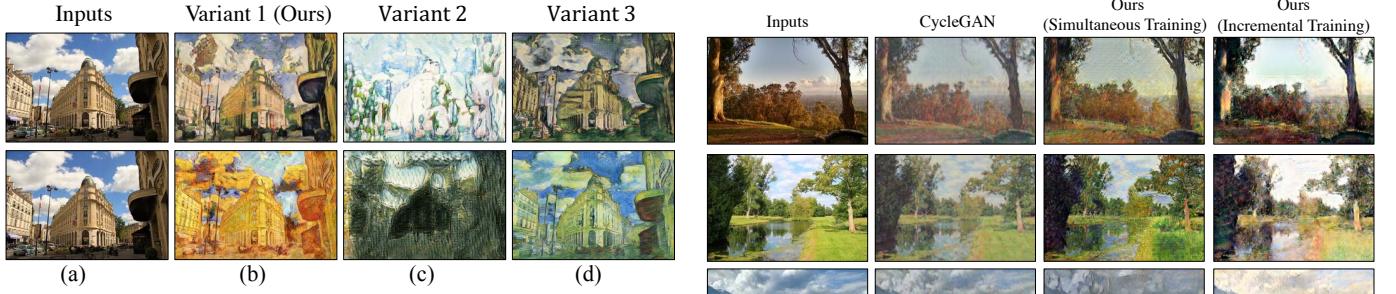


Fig. 15. Qualitative comparison of the influence of different network structures. The first row is the results of photo  $\rightarrow$  Cezanne, and the second row is the results of photo  $\rightarrow$  Van Gogh.

64.14 of baseline CycleGAN, the incremental training achieves a competitive result.

#### G. Linear Interpolation of Styles

Since our proposed model achieves multi-collection style transfer by switching gates  $c$  to different branches  $T(Enc(x), c)$ , we can blend multiple styles by adjusting the gate weights to create a new style or generate transitions between styles of different artists or genres:

$$\tilde{G}(x, c_1, c_2) = Dec(\alpha \cdot T(Enc(x), c_1) + (1 - \alpha) \cdot T(Enc(x), c_2)) \quad (13)$$

where  $c_1$  and  $c_2$  indicate the gates corresponding to different style branches, and  $\alpha$  indicates the weight for convex combination of styles. In Figure 17, we show an example of interpolation from Monet to Van Gogh with the trained model as we vary  $\alpha$  from 0 to 1. The convex combination produces a smooth transition from one style to the other.

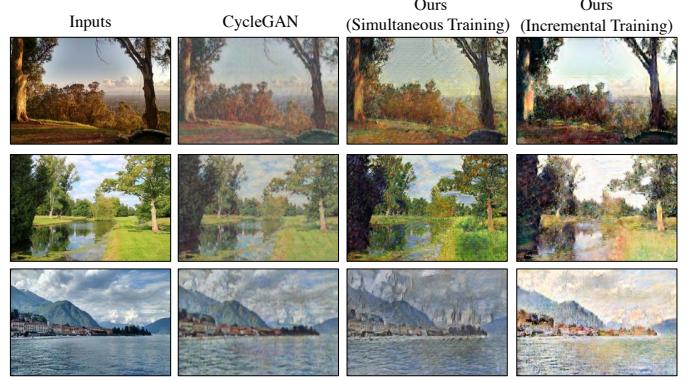


Fig. 16. Comparison of incremental training. From left to right: original inputs, results of CycleGAN [14], results of our methods that all the styles are trained simultaneously, results of incremental training.

## VI. CONCLUSIONS AND FUTURE WORK

In this paper, we study multi-collection style transfer in a single network using adversarial training. To integrate styles into a single network, we design a gated network that filters in different network branches with respect to different styles. To learn multiple styles simultaneously, a discriminator and an auxiliary classifier distinguish authentic artworks and their styles. To stabilize GAN training, we introduce the auto-encoder reconstruction loss. Furthermore, the gated transformer module provides the opportunity to explore new styles by assigning different weights to the gates. Experiments demonstrate the stability, functionality, and effectiveness of our model and produce satisfactory results compared with a state-of-art algorithm, in which one network merely outputs images in one style. In the future, we will apply our model to train other conditional image generation tasks (e.g., object transfiguration, season transfer, photo enhancement) and explore to generate diversified style transfer results.



Fig. 17. Style interpolation. The leftmost image is generated in Monet’s style, and the rightmost image is generated in Van Gogh’s style. Images in the middle are convex combinations of the two styles.

## REFERENCES

- [1] A. A. Efros and T. K. Leung, “Texture synthesis by non-parametric sampling,” in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1033–1038.
- [2] H. Lee, S. Seo, S. Ryoo, and K. Yoon, “Directional texture transfer,” in *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2010, pp. 43–48.
- [3] N. Ashikhmin, “Fast texture transfer,” *IEEE Computer Graphics and Applications*, vol. 23, no. 4, pp. 38–43, 2003.
- [4] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 327–340.
- [5] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.
- [6] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.
- [7] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images.” in *ICML*, 2016, pp. 1349–1357.
- [8] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, “Stylebank: An explicit representation for neural image style transfer,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [9] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, “Diversified texture synthesis with feed-forward networks,” *arXiv preprint arXiv:1703.01664*, 2017.
- [10] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017*, 2017, pp. 2642–2651.
- [11] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [12] F. Luan, S. Paris, E. Shechtman, and K. Bala, “Deep photo style transfer,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [13] 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*, 2014, pp. 2672–2680.
- [14] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [15] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, “Learning to discover cross-domain relations with generative adversarial networks,” *arXiv preprint arXiv:1703.05192*, 2017.
- [16] Z. Yi, H. Zhang, P. T. Gong *et al.*, “Dualgan: Unsupervised dual learning for image-to-image translation,” *arXiv preprint arXiv:1704.02510*, 2017.
- [17] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [18] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 341–346.
- [19] M. E. Gheche, J. F. Aujol, Y. Berthoumieu, and C. A. Deledalle, “Texture reconstruction guided by a high-resolution patch,” *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 549–560, Feb 2017.
- [20] A. Akl, C. Yaacoub, M. Donias, J. P. D. Costa, and C. Germain, “Texture synthesis using the structure tensor,” *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 4082–4095, Nov 2015.
- [21] M. Elad and P. Milanfar, “Style transfer via texture synthesis,” *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2338–2351, 2017.
- [22] O. Frigo, N. Sabater, J. Delon, and P. Hellier, “Split and match: Example-based adaptive patch sampling for unsupervised style transfer,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 553–561.
- [23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
- [24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [25] A. Mahendran and A. Vedaldi, “Understanding deep image representations by inverting them,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.
- [26] C. Li and M. Wand, “Combining markov random fields and convolutional neural networks for image synthesis,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2479–2486.
- [27] A. Selim, M. Elgharib, and L. Doyle, “Painting style transfer for head portraits using convolutional neural networks,” *ACM Transactions on Graphics (ToG)*, vol. 35, no. 4, p. 129, 2016.
- [28] D. Vincent, S. Jonathon, and K. Manjunath, “A learned representation for artistic style,” April 2017.
- [29] H. Xun and B. Serge, “Arbitrary style transfer in real-time with adaptive instance normalization,” April 2017.
- [30] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, “Universal style transfer via feature transforms,” in *Advances in Neural Information Processing Systems*, 2017, pp. 385–395.
- [31] C. Li and M. Wand, “Precomputed real-time texture synthesis with markovian generative adversarial networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 702–716.
- [32] N. Jetchev, U. Bergmann, and R. Vollgraf, “Texture synthesis with spatial generative adversarial networks,” *arXiv preprint arXiv:1611.08207*, 2016.
- [33] U. Bergmann, N. Jetchev, and R. Vollgraf, “Learning texture manifolds with the periodic spatial gan,” in *Thirty-fourth International Conference on Machine Learning (ICML)*, 2017.
- [34] W. Lotter, G. Kreiman, and D. Cox, “Unsupervised learning of visual structure using predictive generative networks,” *arXiv preprint arXiv:1511.06380*, 2015.
- [35] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” *arXiv preprint arXiv:1609.04802*, 2016.
- [36] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [37] Y. Taigman, A. Polyak, and L. Wolf, “Unsupervised cross-domain image generation,” April 2017.

- [38] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
- [39] A. Odena, "Semi-supervised learning with generative adversarial networks," *arXiv preprint arXiv:1606.01583*, 2016.
- [40] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.
- [41] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.
- [42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [43] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [44] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.
- [45] Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang, J. Wang, and Y. Yu, "Activation maximization generative adversarial nets," in *International Conference on Learning Representations*, 2018.
- [46] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.
- [47] H. A. Aly and E. Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1647–1659, 2005.
- [48] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [49] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [50] 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*, 2017, pp. 6629–6640.