Line Drawings for Face Portraits From Photos Using Global and Local Structure Based GANs

Ran Yi, Mengfei Xia, Yong-Jin Liu, *Senior Member, IEEE*, Yu-Kun Lai, *Member, IEEE*, and Paul L. Rosin, *Member, IEEE*

Abstract—Despite significant effort and notable success of neural style transfer, it remains challenging for highly abstract styles, in particular line drawings. In this paper, we propose APDrawingGAN++, a generative adversarial network (GAN) for transforming face photos to artistic portrait drawings (APDrawings), which addresses substantial challenges including highly abstract style, different drawing techniques for different facial features, and high perceptual sensitivity to artifacts. To address these, we propose a composite GAN architecture that consists of local networks (to learn effective representations for specific facial features) and a global network (to capture the overall content). We provide a theoretical explanation for the necessity of this composite GAN structure by proving that any GAN with a single generator cannot generate artistic styles like APDrawings. We further introduce a classification-and-synthesis approach for lips and hair where different drawing styles are used by artists, which applies suitable styles for a given input. To capture the highly abstract art form inherent in APDrawings, we address two challenging operations—(1) coping with lines with small misalignments while penalizing large discrepancy and (2) generating more continuous lines—by introducing two novel loss terms: one is a novel distance transform loss with nonlinear mapping and the other is a novel line continuity loss, both of which improve the line quality. We also develop dedicated data augmentation and pre-training to further improve results. Extensive experiments, including a user study, show that our method outperforms state-of-the-art methods, both qualitatively and quantitatively.

Index Terms—Face portrait, style transfer, image translation, generative adversarial network

1 Introduction

Drawing is the artist's most direct and spontaneous expression. A species of writing: it reveals, better than does painting, his true personality.

Edgar Degas

Line drawings for face portraits are a longstanding and distinct art form, which typically use a sparse set of continuous graphical elements (e.g., lines) to capture the distinctive appearance of a person. They are drawn in the presence of the person or their photo, and rely on a holistic approach of observation, analysis and experience. An artistic portrait drawing should ideally capture the personality and the feelings of the person. Even for an artist with professional training, it usually requires several hours to finish a good portrait.

Training a computer program with artists' drawings and automatically transforming an input photo into high-quality artistic drawings is highly desired. In particular, with the development of deep learning, *neural style transfer* (NST)—which uses CNNs to perform image style transfer—was proposed [1]. Later on, *generative adversarial network* (GAN) based style transfer methods (e.g., [2], [3], [4], [5]) have achieved especially good results, by utilizing sets of (paired or unpaired) photos and stylized images for learning. These existing methods are mostly demonstrated using cluttered styles, which contain many fragmented graphical elements such as brush strokes, and have a significantly lower requirement for the quality of individual elements (i.e., imperfections are much less noticeable).

Artistic portrait drawings (APDrawings) are substantially different in style from portrait painting styles studied in previous work, mainly due to the following five aspects. First, the APDrawing style is highly abstract, containing a small number of sparse but continuous graphical elements. Defects (such as extra, missing or erroneous lines) in APDrawings are much more visible than other styles such as paintings (e.g., impressionist and oil painting) involving a dense collection of thousands of strokes of varying sizes and shapes. Second, there are stronger semantic constraints for APDrawing style transfer than for general style transfer. In particular, facial features should not be missing or displaced. Even small artifacts (e.g., around the eye) can be clearly visible, distracting and unacceptable. Third, the rendering in APDrawings is not consistent between different facial parts (e.g., eyes versus hair). Fourth, the elements (e.g., the outline of facial parts) in APDrawings are not precisely located by artists, posing a challenge for methods based on pixel correspondence (e.g., Pix2Pix [2]). Finally, artists put lines in APDrawings that are not directly related to low level features in the view or photograph of the person. Examples include lines in the hair indicating the flow, or lines indicating the presence of facial features even if the

- *R. Yi, M. Xia, and Y.-J. Liu are with BNRist, MOE-Key Laboratory of Pervasive Computing, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: {yr16, xiamf16}@mails.tsinghua.edu.cn, liuyongjin@tsinghua.edu.cn.*
- *Y.-K. Lai and P.L. Rosin are with the School of Computer Science and Informatics, Cardiff University, CF24 3AA Cardiff, UK. E-mail: {laiy4, rosinpl}@cardiff.ac.uk.*

Manuscript received 30 June 2019; revised 2 Apr. 2020; accepted 7 Apr. 2020. Date of publication 16 Apr. 2020; date of current version 2 Sept. 2021. (Corresponding author: Yong-Jin Liu.) Recommended for acceptance by E. Shechtman. Digital Object Identifier no. 10.1109/TPAMI.2020.2987931

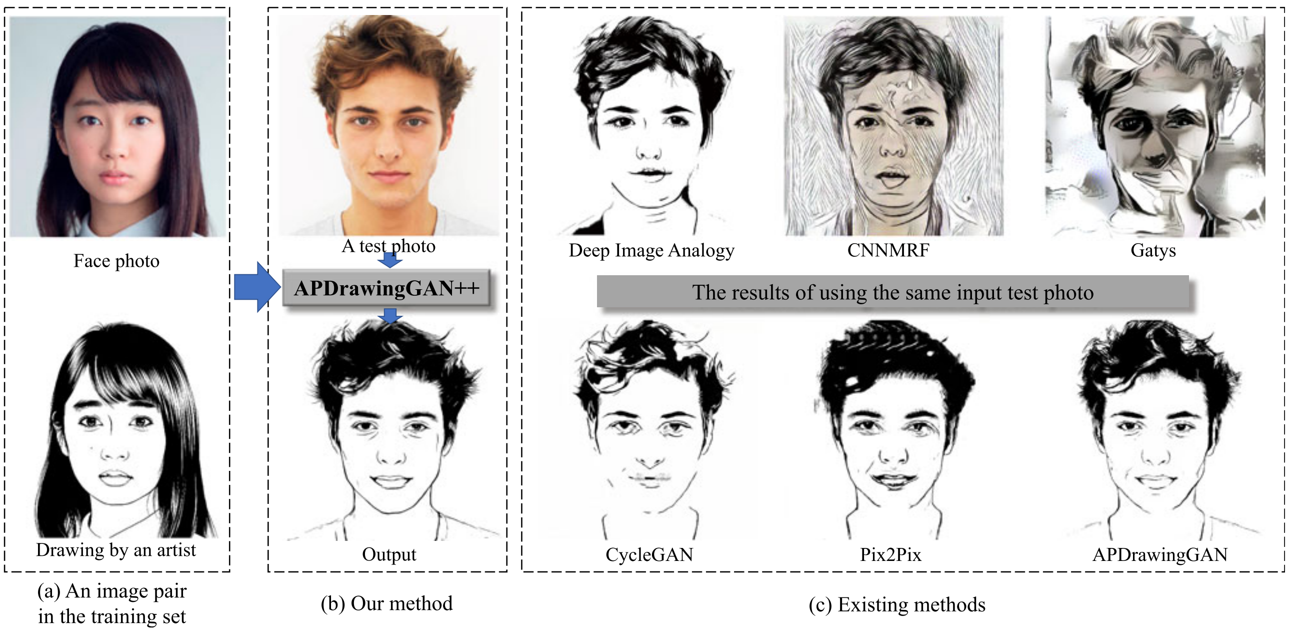


Fig. 1. (a) An artist draws a portrait drawing using a sparse set of lines and very few shaded regions to capture the distinctive appearance of a person or a given face photo. (b) Our APDrawingGAN++ learns this artistic drawing style and automatically transforms a face photo into a high-quality artistic portrait drawing. (c) Using the same input face photo, six state-of-the-art style transfer methods cannot generate desired artistic drawings. The drawings output by Deep Image Analogy [6], CNNMRF [7] and Gatys [1] seem not to obtain the right style and have some facial features changed, which makes them difficult to recognize. CycleGAN [3] and Pix2Pix [2] produce false details around hairs, eyes or corners of the lip. APDrawingGAN [8] works relatively well, but produces less delicate facial features (e.g., more messy lips) and hair than our result. More results are presented in Section 7 and appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2020.2987931.

image contains no discontinuities. Such elements of the drawings are hard to learn. Therefore, even state-of-the-art image style transfer algorithms (e.g., [1], [2], [3], [6], [7], [9]) often fail to produce good and expressive APDrawings. See Fig. 1c for some examples.

To address the above challenges, in our previous conference work [8], we proposed a composite GAN architecture called APDrawingGAN to generate APDrawings from face photos. To learn different drawing styles for different facial regions, APDrawingGAN involves several local networks dedicated to face structure, along with a global network to capture holistic characteristics. To further cope with the line-stroke-based style in artists' drawings, APDrawingGAN applies a novel distance transform (DT) loss to learn stroke lines in APDrawings. APDrawingGAN works well when its output APDrawings are roughly examined as a whole. However, the subtle facial features and hair are still far from the masterful level achieved by professional artists (see Figs. 1a and 1c for an example).

In this paper, we substantially improve upon our previous work [8] and propose APDrawingGAN++, which can generate masterful APDrawings by learning delicate facial features (Fig. 1b). In particular, we make the following contributions:

- *Network structure*. Face photos and APDrawings are two quite different forms. APDrawingGAN++ uses two levels to transform between them: a coarse level (which makes use of CNNs with residual blocks to transform a face photo into a rough APDrawing form) and a fine level (which makes use of auto-encoders to fine tune a rough APDrawing into a masterful one). APDrawingGAN [8] only uses the coarse level, while APDrawingGAN++ furthermore introduces local auto-encoders into the fine level. Notably, we provide a theoretical explanation of using multiple generators for APDrawing generation. The theoretical proof can also shed light on other tasks using multiple networks for image generation.
- *Loss function*. To capture the highly abstract art form inherent in APDrawings, we propose a loss function dedicated to APDrawing with five loss terms, including a novel DT loss (to promote line-stroke based style in APDrawings), a novel line continuity loss (to enhance line continuity in APDrawings) and a local transfer loss (for local networks to preserve facial features). Unlike APDrawingGAN, APDrawingGAN++ introduces the new line continuity loss and applies a nonlinear mapping to the DT before computing the DT loss, so that the resulting APDrawings are kept cleaner by strongly penalizing large misalignments while tolerating small misalignments, which are typically present in artist drawings.
- *Training scheme*. We pre-train our model using 6,655 frontal face photos collected from ten face datasets, and construct an APDrawing dataset (containing 140 high-resolution face photos and corresponding portrait drawings by a professional artist) suitable for training and testing. Since different skin colors are unbalanced within the data set, advancing from APDrawingGAN, APDrawingGAN++ applies histogram matching to provide data augmentation for the training set. This changes the face color in the photos, enabling APDrawingGAN++ to achieve better

results for dark faces that are under-represented in the training set.

Extensive experiments and a user study demonstrate that APDrawingGAN++ produces significantly better artistic drawings than state-of-the-art methods. Especially, APDrawingGAN++ outperforms APDrawingGAN with improved expressive facial features. We developed and released a mini-program on WeChat, which is the most popular free messaging and calling app in China. Our mini-program became popular, receiving about 400K user clicks in only two weeks. The source code of APDrawingGAN++ is available.[1]

2 Related Work

Image stylization has been widely studied in non-photorealistic rendering and deep learning research. Below we summarize related work in three aspects.

2.1 Style Transfer Using Neural Networks

Gatys *et al.* [1] first proposed an NST method using a CNN to transfer the stylistic characteristics of a style image to a content image. For a given image, its content and style features are represented by high layer features and texture information captured by Gram matrices [10] in a VGG network, respectively. Style transfer is achieved by optimizing an image to match both the content of the content image and the style of the style image. This method performs well on oil painting style transfer of various artists. However, their style is modeled as texture features, and thus not suitable for our target style with little texture.

Li and Wand [7] used a Markov Random Field (MRF) loss instead of the Gram matrix to encode the style, and proposed the combined MRF and CNN model (CNNMRF). CNNMRF can be applied in both non-photorealistic (artwork) and photo-realistic image synthesis, since local patch matching is used in MRF loss and promotes local plausibility. However, local patch matching restricts this method to only work well when the style and content images contain elements of similar local features.

Liao *et al.* [6] proposed Deep Image Analogy for visual attribute transfer by finding semantically meaningful dense correspondences between two input images. They compute correspondence between feature maps extracted by a CNN. Deep Image Analogy was successfully applied to photo-to-style transfer, but when transferring APDrawing style, image content is sometimes affected, making subjects in the resulting images less recognizable.

Johnson *et al.* [11] proposed the concept of perceptual-loss-based high-level features and trained a feed forward network for image style transfer. Similar to [1], their texture-based loss function is not suitable for our style.

In addition to aforementioned limitations for APDrawing style transfer, most existing methods require the style image to be close to the content image.

2.2 Non-Photorealistic Rendering of Portraits

Non-photorealistic rendering (NPR) is a branch in computer graphics that targets creating expressive rendering styles [12] often in (but not constrained to) traditional artistic styles, and involves abstraction, recoloring, simulation of various media, etc. Because of the importance of portraits, there are many NPR methods developed specifically for dealing with portraits [13]. Similar to general NPR methods, NPR portrait methods can be categorized into stroke-based methods [14], [15], [16], region-based methods [17], [18], [19], [20] and texture-transfer-based methods [21], [22]. Stroke-based methods render portraits by simulating the stroke placement of artists. Some methods [14] warp strokes from a training artistic image, while others [15], [16] learn a stroke style model by observing stroke characteristics from artist strokes. Region-based methods decompose the input image into components, and then either match them with templates from training artistic images and recombine matched templates into a portrait [17], [18], [19], or render different facial components using different specialized algorithms [20]. Texture-transfer-based methods transfer textures from the exemplar artistic image to the output rendering, using parametric texture synthesis [21] or non-parametric texture synthesis [22].

Among these methods, sketch or line drawing rendering methods [16], [17], [23], [24] are more relevant to our problem. However, the target APDrawing style is different from these styles in having smaller shape deformation, little shading, and more delicate lines. The target style is challenging because it contains sparse but delicate lines, which are hard to render via existing NPR algorithms, and therefore we exploit deep-learning based methods to provide a solution.

2.3 GAN-Based Image Synthesis

Generative Adversarial Networks (GANs) [25] have achieved much progress in solving many image synthesis problems, in which closely related to our work are Pix2Pix and CycleGAN.

Pix2Pix [2] is a general framework for image-to-image translation, which explores GANs in a conditional setting [26]. Pix2Pix can be applied to a variety of image translation tasks and achieves impressive results on various tasks including semantic segmentation, colorization and sketch to photo translation, etc.

CycleGAN [3] is designed to learn translation between two domains without paired data by introducing cycle-consistency loss. This model is particularly suitable for tasks in which paired training data is not available. When applied to a dataset with paired data, this method produces results similar to the fully supervised Pix2Pix, but with much more training time.

Neither Pix2Pix nor CycleGAN works well for APDrawing styles and often generates blurry or messy results due to the five challenges summarized in Section 1 for APDrawings.

3 Overview of APDrawingGAN++

We model the process of learning to transform face photos to APDrawings as a function Ψ which maps the face photo domain $\mathcal{P}$ into a black-and-white line-stroke-based APDrawing domain $\mathcal{A}$. The function Ψ is learned from paired training data $S_{data} = \{(p_i, a_i) | p_i \in \mathcal{P}, a_i \in \mathcal{A}, i = 1, 2, \ldots, N\}$, where N is the number of photo-APDrawing pairs in the training set.

1. https://github.com/yiranran/APDrawingGAN2

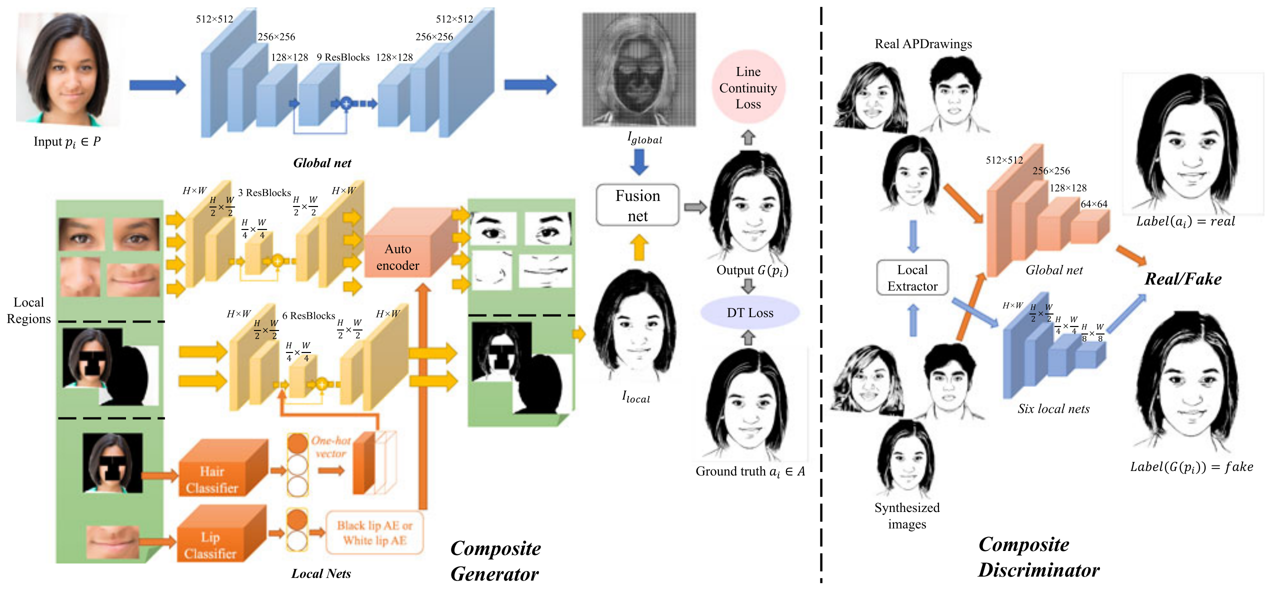


Fig. 2. The framework of the proposed APDrawingGAN++. The composite generator G takes a face photo $p_i \in \mathcal{P}$ as input and consists of a global network (for global facial structure), six local networks (for four local facial regions, the hair, and the background region), five auto-encoders, two classifiers and a fusion network. Outputs of the auto-encoders and hair/background generators are combined into I_{local} and fused with the output I_{global} of the global network to generate the final output $G(p_i)$. The loss function includes five terms, in which a novel DT loss is introduced to better learn delicate artistic line styles, and a novel line continuity loss is introduced to generate more continuous lines. The composite discriminator D distinguishes whether the input is a real APDrawing or not based on the classification results by combining both a global discriminator and six local discriminators.

Our model is based on the GAN framework, which consists of a composite generator G and a composite discriminator D, both of which are CNNs specifically designed for APDrawings with line-stroke-based artist drawing style. The generator G learns to output an APDrawing in $\mathcal{A}$ while the discriminator D learns to determine whether an image is a real APDrawing or generated.

Since our model is based on GANs, the discriminator D is trained to maximize the probability of assigning the correct label to both real APDrawings $a_i \in \mathcal{A}$ and synthesized drawings $G(p_i)$, $p_i \in \mathcal{P}$, and simultaneously G is trained to minimize this probability. Denote the loss function as $L(G,D)$, which is specially designed to include five terms $L_{adv}(G,D)$, $L_{\mathcal{L}_1}(G,D)$, $L_{DT}(G,D)$, $L_{conti}(G,D)$ and $L_{local}(G,D)$. Then the function Ψ can be formulated by solving the following min-max problem with the function $L(G,D)$:

$$\begin{aligned}\min_G \max_D L(G,D) &= L_{adv}(G,D) + \lambda_1 L_{\mathcal{L}_1}(G,D)\\ &+ \lambda_2 L_{DT}(G,D) + \lambda_3 L_{local}(G,D) + \lambda_4 L_{conti}(G,D).\end{aligned} \tag{1}$$

In Section 4, we introduce the architecture of APDrawingGAN++. Sections 5 and 6 present the five terms in $L(G,D)$ and the training scheme, respectively. Section 7 presents a comprehensive evaluation of APDrawingGAN++ and compares it with state of the art. Finally Section 8 offers the concluding remarks. An overview of APDrawingGAN++ is illustrated in Fig. 2.

4 Network Architecture

We propose a composite structure for both generator and discriminator, each of which includes a global network and six local networks. The six local networks correspond to the local facial regions of the left eye, right eye, nose, lip, hair and the background. Furthermore, the generator has an additional fusion network to synthesize the artistic drawings from the output of global and local networks. The reason behind this composite structure is that in portrait drawing, artists adopt different drawing techniques for different parts of the face. For example, fine details are often drawn for eyes, and curves drawn for hair usually follow the flow of hair but do not precisely correspond to image intensities. Since a single CNN shares filters across all locations in an image and is very difficult to encode/decode multiple drawing features, the design of composite global and local networks with multiple CNNs can help the model better learn facial features in different locations.

Different from APDrawingGAN [8], we introduce auto-encoders into the generator network to further improve local fine details in APDrawings. To cope with six combinations of two lip styles and three hair styles, we introduce two classifiers for lips and hair respectively, and the output of the classifiers is used to guide the local generators/auto-encoders towards one of the target styles, while avoiding synthesizing output in an undesirable intermediate style.

4.1 Composite Generator G

The generator G transforms input face photos to APDrawings. The style of APDrawings is learned once the model is trained. In the composition of $G = \{G_{global}, G_{l*}, E_*, C_*, G_{fusion}\}$, G_{global} is a global generator, $G_{l*} = \{G_{l\_eye\_l}, G_{l\_eye\_r}, G_{l\_nose}, G_{l\_lip}, G_{l\_hair}, G_{l\_bg}\}$ is a set of six local generators, $E_* = \{E_{eye\_l}, E_{eye\_r}, E_{nose}, E_{lip\_b}, E_{lip\_w}\}$ is a set of five auto-encoders, $C_* = \{C_{lip}, C_{hair}\}$ is a set of two classifiers and G_{fusion} is a fusion network. We design G using CNNs with residual blocks [27].

4.1.1 Local Generators

Each of $G_{l\_eye\_l}$, $G_{l\_eye\_r}$, G_{l_nose} and G_{l_lip} is a CNN with a flat-convolution layer, two down-convolution layers, three residual blocks, two up-convolution layers and one final flat-convolution layer. Each of G_{l_hair} and G_{l_bg} is a CNN with a flat-convolution layer, two down-convolution layers, six residual blocks, two up-convolution layer and one final flat-convolution layer. The role of local generators in G_{l*} is to learn the transform ability (from photos to APDrawings) of different local face features; e.g., transforming hair photo to a hairy style (i.e., repeated wispy details by short choppy or long strokes to capture the soft wispiness of individual hair strands), eyes and nose to line style, and lip to solid or line style.

The inputs to $G_{l\_eye\_l}$, $G_{l\_eye\_r}$, G_{l_nose}, G_{l_lip} are local regions in the photo, centered at the facial landmarks (i.e., left eye, right eye, nose and lip) obtained by the MTCNN model [28]. The input to G_{l_bg} is the photo background region detected by a portrait segmentation method [29]. The input to G_{l_hair} is the remaining region in the face photo.

The success of APDrawingGAN [8] demonstrates that the six local generators G_{l*}, together with G_{global} and G_{fusion}, have a powerful ability to transform a face photo (i.e., color pixel information) into an APDrawing (i.e., highly abstract, imprecise, sparse but continuous graphical elements). However, the subtle facial features and hair in the output of G are still far from artistically beautiful. APDrawingGAN++ uses G_{l*} to transform different portions of the input photo into a rough APDrawing at a coarse level. Then at a fine level, APDrawingGAN++ uses two lip and hair classifiers for specifying the drawing style for different facial features (Section 4.1.2) and applies trained auto-encoders to transform rough APDrawing portions into masterful APDrawing portions (Section 4.1.3), which are naturally fused together using G_{global} and G_{fusion} (Section 4.1.4).

We note that using multiple local GANs for different facial regions is not a new idea (e.g., [30]). Our contribution lies in the new composite network structure that efficiently incorporates the set of six local generators G_{l*} with G_{global} and G_{fusion} as an integrated framework. In addition to its practical value for the challenging APDrawing style, we also provide a theoretical explanation in Section A2 of the appendix for this composite structure, available in the online supplemental material.

4.1.2 Style-Control Classifiers for Lips and Hair

The APDrawings for lips and hair in our APDrawing dataset exhibit multiple styles, e.g., there are both full black lip drawings and full white (outline) lip drawings (Fig. 3). Artists tend to draw lips with lipstick in the full black manner, and other lips in the white manner. Learning to distinguish both styles using the same network is challenging and often produces suboptimal drawings. To better generate lip and hair drawings, we introduce two classifiers C_{lip}, C_{hair} to detect the target style for the lip and hair regions respectively. Then we use the detected class information to guide the generation towards the desired style.

The inputs to C_{lip} and C_{hair} are the local lip and hair regions of the face photo, respectively, and their outputs are class labels. The ground truth class labels for face photos are assigned according to the paired artist drawings. Lip regions are classified into white lips and black lips, and hair regions are classified into light hair, dark hair and middle hair (based on the proportion of white lines). Some examples are illustrated in Fig. 3. Classifiers adopt an architecture with m convolution layers followed by a fully-connected

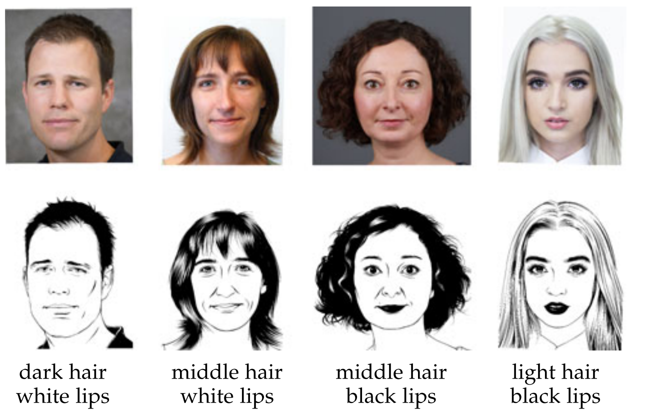


Fig. 3. Some examples of different types of hair (dark/middle/light) and lips (white/black). First row: face photos. Second row: artist drawings. The ground truth class labels for each face photo are assigned according to the paired artist drawing.

layer ($m = 4$ for lip regions and $m = 7$ for hair regions). C_{lip} and C_{hair} are trained using the cross entropy loss.

To guide the local generators/auto-encoders towards a desired style, the output of C_{lip} is used to control the auto-encoder enhancement (to be detailed in Section 4.1.3) and the output of C_{hair} is used as follows. Similar to Im2Pencil [31], which uses a one-hot vector to represent different outline/shading styles, we use a one-hot vector to represent different drawing styles of hair regions. The output of C_{hair} is converted to a 3-dimensional one-hot vector, then mapped to a 3-channel map, and concatenated with the feature extracted by the encoder of the hair local generator G_{l_hair} (concatenated after the second down-convolution layer). The concatenated feature is then decoded by G_{l_hair}'s decoder. Therefore, the output of G_{l_hair} is controlled by the hair style detected by C_{hair}.

4.1.3 Auto-Encoders for Subtle Facial Features

After the local generators $G_{l\_eye\_l}$, $G_{l\_eye\_r}$, G_{l_nose}, G_{l_lip} output rough APDrawings for different facial parts, we introduce auto-encoders E_{eye_l}, E_{eye_r}, E_{nose}, $E_{lip\_b/w}$ to improve local drawings. Both the input and output of these auto-encoders are parts of APDrawings. By training on the APDrawing dataset, each auto-encoder learns a good feature representation (for an encoder) and a good decoder (to reconstruct high-quality APDrawings of the corresponding facial part). In APDrawingGAN++, we feed synthesized, rough APDrawings output from local generators into auto-encoders. The auto-encoders encode rough APDrawings into concise feature representations, which are in turn decoded for reconstructing high-quality APDrawings close to the artist drawings.

The auto-encoders adopt an architecture of four down convolution layers, two fully-connected layers and four up convolution layers. In preprocessing, we use the APDrawing dataset to train the auto-encoders and choose the mean square error (MSE) loss that is computed in a compact mask m_c (ref. Fig. 4c) obtained by OpenFace [32]. In APDrawingGAN++, these auto-encoders are fixed during GAN training. The output of the auto-encoders r_{out} is then combined with input r_{in} using the compact mask m_c, i.e., the combined result $r_{in} \cdot m_c + r_{out} \cdot (1 - m_c)$ is used as the refined

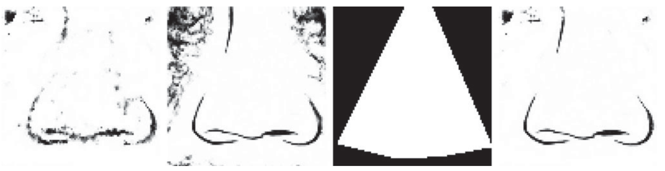


Fig. 4. An example of the nose auto-encoder's input and the reconstructed nose drawing. (a) synthesized nose drawing by the local generator G_{l_nose}, (b) reconstructed nose drawing by nose auto-encoder E_{nose}, (c) the compact mask for the nose, and (d) refined nose drawing by combining input and output of E_{nose} using the compact mask.

local drawing. Fig. 4 shows an example, where the refined nose (Fig. 4d) is depicted by much nicer lines than the synthesized, rough nose (Fig. 4a) output from G_{l_nose}. Such auto-encoders also benefit from our composite GAN architecture, as learning individual facial features is much easier than learning the entire face drawings as a whole, making it possible to train these auto-encoders with a limited number of training examples.

For the lip region, two special auto-encoders E_{lip_b} and E_{lip_w} are trained separately. E_{lip_b} is dedicated to black lip generation while E_{lip_w} is dedicated to white lip generation. Which lip auto-encoder to use is determined by the output of the lip classifier C_{lip}.

Finally, we blend outputs of auto-encoders and outputs of hair and background generators into an aggregated drawing I_{local}, by using min pooling in overlapping regions. This min pooling operation can effectively retain responses from individual local generators, as low intensities are treated as responses for black pixels in artistic drawings. To avoid sharp intensity change at the boundaries of local regions during blending, we adopt a *soft border mask* for each local region.

4.1.4 Global Generator and Fusion Network

G_{global} is a CNN with a flat-convolution layer, two down-convolution layers, nine residual blocks, two up-convolution layers and one final flat-convolution layer. G_{fusion} consists of a flat convolution layer, two residual blocks and a final convolution layer. We use G_{fusion} to fuse together I_{local} and I_{global} (i.e., the output of G_{global}) for obtaining the final high-quality APDrawing. In many previous GAN models (e.g., [25], [33]), usually some noise is input or added in the generator network. Following [2], we do not add noise in G explicitly, but use dropout [34] in residual blocks to work as noise.

Different from the multiple GAN structure proposed in [30], our composite GAN model contains the global generator and fusion network. They play an important role in APDrawing style generation and we present an ablation study in Section A6.3 of the appendix to demonstrate its effectiveness, available in the online supplemental material.

4.2 Composite Discriminator D

The discriminator D distinguishes whether the input drawing is a real artist's portrait drawing or not. In the hierarchy of $D = \{D_{global}, D_{l*}\}$, D_{global} is a global discriminator and $D_{l*} = \{D_{l\_eye\_l}, D_{l\_eye\_r}, D_{l\_nose}, D_{l\_lip}, D_{l\_hair}, D_{l\_bg}\}$ is a set of six local discriminators. D_{global} examines the whole drawing to judge the holistic APDrawing features, while the local discriminators in D_{l*} examine different local regions to evaluate the quality of fine details.

We implement D_{global} and all local discriminators in D_{l*} using the Markovian discriminator in Pix2Pix [2]. The only difference is the input: the whole drawings or different local regions. The Markovian discriminator processes each 70×70 patch in the input image and examines the style of each patch. Local patches from different granularities (i.e., coarse and fine levels at global and local input) allow the discriminator to learn local patterns and better discriminate real artists' drawings from synthesized drawings.

5 Loss Function

There are five terms in the loss function in Eq. (1), which are explained as follows.

Adversarial loss L_{adv} models the discriminator's ability to correctly distinguish real or false APDrawings. Following Pix2Pix [2], the adversarial loss is formulated as

$$\begin{aligned} L_{adv}(G, D) = \sum_{D_j \in D} \mathbb{E}_{(p_i, a_i) \sim S_{data}} [\log (D_j(p_i, a_i) \\ + \log (1 - D_j(p_i, G(p_i)))]. \end{aligned} \tag{2}$$

When $D_j \in D_{l*}$, the images p_i, a_i and $G(p_i)$ are all restricted to the local region specified by D_j. As D maximizes this loss while G minimizing it, L_{adv} forces the synthesized drawings to become closer to the target domain $\mathcal{A}$.

Pixel-wise loss $L_{\mathcal{L}_1}$ drives the synthesized drawings close to ground-truth drawings in a pixel-wise manner. We compute the $L_{\mathcal{L}_1}$ loss for each pixel in the whole drawing

$$L_{\mathcal{L}_1}(G, D) = \mathbb{E}_{(p_i, a_i) \sim S_{data}} [\| G(p_i) - a_i \|_1]. \tag{3}$$

Using the $\mathcal{L}_1$ norm generally outputs less blurry results than the $\mathcal{L}_2$ norm and so is more suitable for APDrawing style.

Line-promoting distance transform loss with nonlinear mapping L_{DT} is a novel measure specially designed for promoting line strokes in the style of APDrawings. Since the elements in APDrawings are not located precisely corresponding to image intensities, we introduce L_{DT} to tolerate the small misalignments—that are often present in artists' portrait drawings—and to better learn stroke lines in APDrawings. The distance-based loss L_{DT} can tolerate small misalignments but penalizes big misalignments, while $\mathcal{L}_1$ loss treats both as the same. To do so, we make use of distance transform and chamfer matching as follows.

A DT (a.k.a. distance map) can be represented by a digital image, in which each pixel stores a distance value. Given a real or synthesized APDrawing x, we define two DTs of x as images $I_{DT}(x)$ and $\widetilde{I}_{DT}(x)$: assuming $\hat{x}$ is the binarized image of x, each pixel in $I_{DT}(x)$ stores the distance value to its nearest black pixel in $\hat{x}$ and each pixel in $\widetilde{I}_{DT}(x)$ stores the distance value to its nearest white pixel in $\hat{x}$. Fig. 5 shows an example.

Different from APDrawingGAN [8] that directly uses DT to compute chamfer matching distance, we apply a nonlinear mapping to the DT that puts stronger penalty on large misalignments and reduces the penalty on small misalignments. Since small misalignments frequently appear in artist

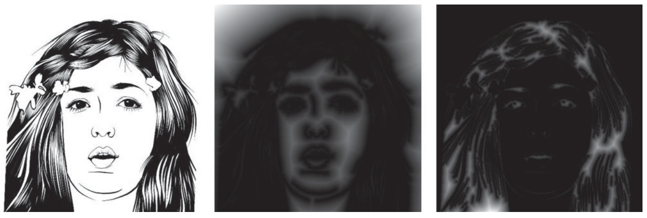


Fig. 5. Distance transforms $I_{DT}(x)$ and $\widetilde{I}_{DT}(x)$ of an APDrawing x.

drawings, the nonlinear mapping can help clean the local messy drawings that are typically output from APDrawing-GAN. We choose the nonlinear mapping as

$$\tilde{I}_{DT}(x)(j,k) = \frac{e^{c \cdot I_{DT}(x)(j,k)} - 1}{e^c - 1}, \tag{4}$$

where $I_{DT}(x)(j,k)$ is the distance value at the pixel (j,k) in $I_{DT}(x)$, c is a parameter to control nonlinearity. We found our method is insensitive to moderate change of this parameter, and it is set to 3.3 in our experiments. Since $I_{DT}(x)$ is originally normalized to the range [0,1], the nonlinear mapping result $\tilde{I}_{DT}(x)$ is also in range [0,1]. $\tilde{I}'_{DT}(x)$ is similarly remapped from $I'_{DT}(x)$.

We train two CNNs[2] to detect black and white lines in APDrawings, denoted as Θ_b and Θ_w. The chamfer matching distance between APDrawings x_1 and x_2 is defined as

$$\begin{aligned} d_{CM}(x_1,x_2) = & \sum_{(j,k)\in\Theta_b(x_1)} \tilde{I}_{DT}(x_2)(j,k) \\ & + \sum_{(j,k)\in\Theta_w(x_1)} \tilde{I}'_{DT}(x_2)(j,k), \end{aligned} \tag{5}$$

where $\tilde{I}_{DT}(x)(j,k)$ and $\tilde{I}'_{DT}(x)(j,k)$ are distance values at the pixel (j,k) in the images $\tilde{I}_{DT}(x)$ and $\tilde{I}'_{DT}(x)$, respectively. $d_{CM}(x_1,x_2)$ measures the sum of remapped distances from each line pixel in x_1 to closest pixel of the same type (black or white) in x_2. Then L_{DT} is defined as

$$\begin{aligned} L_{DT}(G,D) = & \mathbb{E}_{(p_i,a_i)\sim S_{data}}[d_{CM}(a_i, G(p_i)) \\ & + d_{CM}(G(p_i), a_i)]. \end{aligned} \tag{6}$$

Local transfer loss L_{local} puts extra constraints on the intermediate output of six local generators in G_{l*}, and then behaves as a regularization term in the loss function. Denote the six local regions of an APDrawing x as $El(x)$, $Er(x)$, $Ns(x)$, $Mt(x)$, $Hr(x)$ and $Bg(x)$. L_{local} is defined as

$$\begin{aligned} & L_{local}(G,D) = \\ & \mathbb{E}_{(p_i,a_i)\sim S_{data}}\big[||E_{eye\_l}(G_{l\_eye\_l}(El(p_i))) - El(a_i)||_1 \\ & + ||E_{eye\_r}(G_{l\_eye\_r}(Er(p_i))) - Er(a_i)||_1 \\ & + ||E_{nose}(G_{l\_nose}(Ns(p_i))) - Ns(a_i)||_1 \\ & + ||E_{lip\_b/w}(G_{l\_lip}(Mt(p_i))) - Mt(a_i)||_1 \\ & + ||G_{l\_hair}(Hr(p_i), C_{hair}(Hr(p_i))) - Hr(a_i)||_1 \\ & + ||G_{l\_bg}(Bg(p_i)) - Bg(a_i)||_1\big]. \end{aligned} \tag{7}$$

2. We use two-tone NPR images and the corresponding lines generated by the NPR algorithm [23] as data to train the two CNN models.

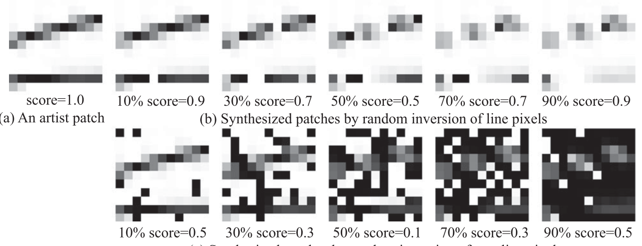


Fig. 6. An example of an artist patch (white-dominant), random inversions of line pixels and non-line pixels and their given line continuity scores.

Line continuity loss L_{conti} is a novel loss term designed for enhancing line continuity. We introduce a line continuity measure by learning from 11×11 patches extracted from artist drawings. We define the line continuity score of artist patches as 1, and give lower scores $(0 \sim 1)$ for synthesized patches obtained by randomly inverting some pixels in artist patches. For a white-dominant artist patch (a patch is white-dominant if there are more white pixels than black pixels; white-dominant patches often contain black lines), we first randomly invert some black pixels (i.e., line pixels) and define the line continuity value of $r\%$ $(0 < r < 100)$ inversion as $0.5 + |r-50|/100$ (i.e., 50 percent inversion gets the lowest continuity score). Then we randomly invert white pixels (i.e., non-line pixels) and define the line continuity value of $r\%$ inversion $0.1 + |r-50|/100$ (refer to Fig. 6). We conduct similar random inversions for black-dominant artist patches.

We train a line continuity prediction network R_{conti} from these artist patches, synthesized (i.e., modified artist) patches and the given continuity scores. The prediction network takes a 11×11 patch as input and outputs a line continuity value. The network contains three flat-convolutions and a fully-connected layer. Then the line continuity score of an APDrawing x is defined as

$$S_{conti}(x) = \mathbb{E}_{\rho_k \sim P(x)} R_{conti}(\rho_k), \tag{8}$$

where $P(x)$ is the set of all patches that are not pure white or pure black, extracted from x, and ρ_k is the kth patch in this set. The higher the line continuity score, the more continuous the lines in APDrawing x. We further define $P_{face}(x)$ as the set of face patches and $P_{nface}(x)$ as the set of non-face patches.

Then line continuity loss L_{conti} is defined as

$$L_{conti}(G,D) = \mathbb{E}_{(p_i,a_i)\sim S_{data}} \mathbb{E}_{\rho_k \sim P(G(p_i))} w_k (1 - R_{conti}(\rho_k)), \tag{9}$$

where weight $w_k = 2$ if $\rho_k \in P_{face}(G(p_i))$, and $w_k = 1$ if $\rho_k \in P_{nface}(G(p_i))$. A higher weight is given to face patches since lines in the face area are often less continuous and harder to learn. Experimental results (to be detailed in Section 7.2.2) show that the novel line continuity loss greatly improves the line continuity and the similarity between synthesized drawings and real drawings.

6 Training APDrawingGAN++

APDrawing Dataset. To train the proposed APDrawingGAN ++, we build a dataset containing 140 pairs of face photos

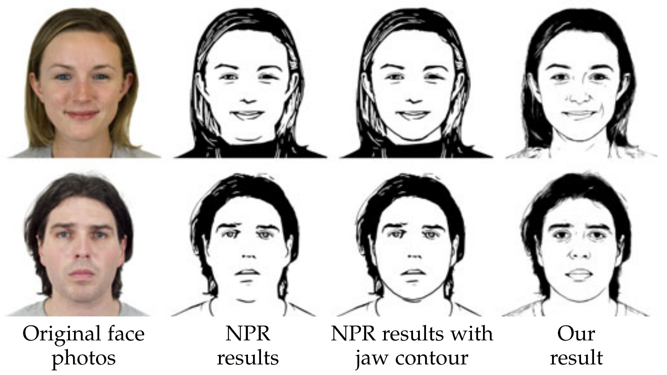


Fig. 7. From left to right: original face photos, NPR results [23], NPR results with clear jaw contours added (used for pre-training), and the results of APDrawingGAN++. Face photos are from the datasets of CFD [35]. Note that these two example images are not used in pre-traning.

and corresponding portrait drawings. To make the training set distribution more consistent, all portrait drawings were drawn by a single professional artist. All images and drawings are aligned and cropped to 512×512 size. Some examples are illustrated in the appendix, available in the online supplemental material.

Initialization With Pre-Training. Since it is time-consuming and laborious for an artist to draw each portrait drawing, our constructed dataset consists of only a small number of image pairs, which makes the training particularly challenging. To address this issue, we use a coarse-level pre-training to make the training starting at a good initial status. We collect 6,655 frontal face photos taken from ten face datasets [35], [36], [37], [38], [39], [40], [41], [42], [43], [44]. For each photo, we generate a synthetic drawing using the two-tone NPR algorithm in [23]. Since it often generates results without clear jaw lines (due to low contrast in the image at these locations), we use the face model in OpenFace [32] to detect the landmarks on the jaws and subsequently add jaw lines to the NPR results. Two examples are illustrated in Fig. 7. Note that the drawings synthesized in this simple way are only a coarse approximation and still far from ideal APDrawings. We use a pre-trained model after 10 epochs as the initialization for the subsequent formal training. Since our NPR generated drawings (unlike artists' drawings) are accurately aligned to the photos, we do not use the distance transform loss in pre-training.

Formal Training. We partition our APDrawing dataset into a training set of 70 image pairs and a test set of 70 image pairs. We first apply data augmentation of small-angle rotation (-$10^o \sim 10^o$) and scaling ($1\sim1.1$) to the training set. Furthermore, we apply the Adam optimizer [45] with learning rate 0.0001, momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$ and batch size of 1.

One more data augmentation that distinguishes APDrawingGAN++ from APDrawingGAN is the *histogram matching augmentation*, which generates face photos with different face colors to further augment the training set. We partition the face photos in the training set into a light face set S_L and a dark face set S_D. For each face photo x in S_L (or S_D), we randomly select one face photo y in S_D (or S_L), and perform a histogram matching on x and y to obtain a matched image $I_{hm}(x,y)$, which has the same color distribution as y while

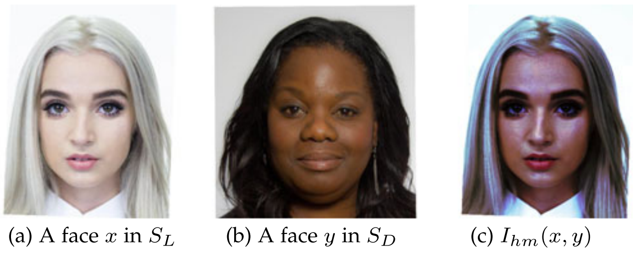


Fig. 8. An example of histogram matching. Histogram matching is performed on a face photo x in S_L and a face photo y in S_D to obtain a histogram matched image $I_{hm}(x,y)$.

preserving the content of x. We use contrast limited histogram matching [46]—which restricts the slope of the mapping function—to avoid extreme and unnatural recolorizations. To avoid any influence from the background, we only match the histogram in face regions for x and y. An example is shown in Fig. 8. The generated new face photo $I_{hm}(x,y)$ is then paired with x's artist drawing to function as a new training sample. We apply histogram matching to each face photo in the training set 7 times using 7 randomly selected images from our APDrawing dataset (to account for existing data augmentation such that after the histogram matching augmentation, the numbers of dark and light skin examples are close). This histogram matching augmentation improves the balance of face colors in the training set and makes the model more generalized and better suited to faces over a wide range of colors.

Auto-Encoder Training. As aforementioned, auto-encoders are trained independently of APDrawingGAN++ and these auto-encoders are fixed during APDrawingGAN++ training. The training of auto-encoders uses the same APDrawing dataset. We also apply the same data augmentation methods (i.e., small-angle rotation and scaling) as APDrawingGAN++ (except for histogram matching augmentation, since the input to auto-encoders are local APDrawings that are irrelevant to the skin color). Furthermore, we apply one more data augmentation of translation, because auto-encoders involve fully-connected layers and are not translation invariant.

As shown in Fig. 7, the NPR results are far from ideal APDrawings and thus can only serve as a coarse approximation in the pre-training phase. In other words, training with only NPR results cannot obtain desired APDrawings. In addition to our two-step training strategy (i.e., pre-training and formal training), another possible strategy is to use the mixed training with both paired and unpaired data (e.g., the strategy in [47]). We compare our training strategy with the mixed training strategy of [47] in Section A7 of the appendix, available in the online supplemental material. The results show that our strategy can obtain better results.

7 Experiments

We implemented APDrawingGAN++ in PyTorch [48] and conducted experiments on a computer with an NVIDIA Titan Xp GPU. The input and output of the generator G are color photos and gray drawings, respectively, and so the numbers of input and output channels are 3 and 1. In all our experiments, the parameters in Eq. (1) are fixed at $\lambda_1 = 100$, $\lambda_2 = 0.35$, $\lambda_3 = 25$ and $\lambda_4 = 40$. All the evaluation results presented in this section are based on the test set to

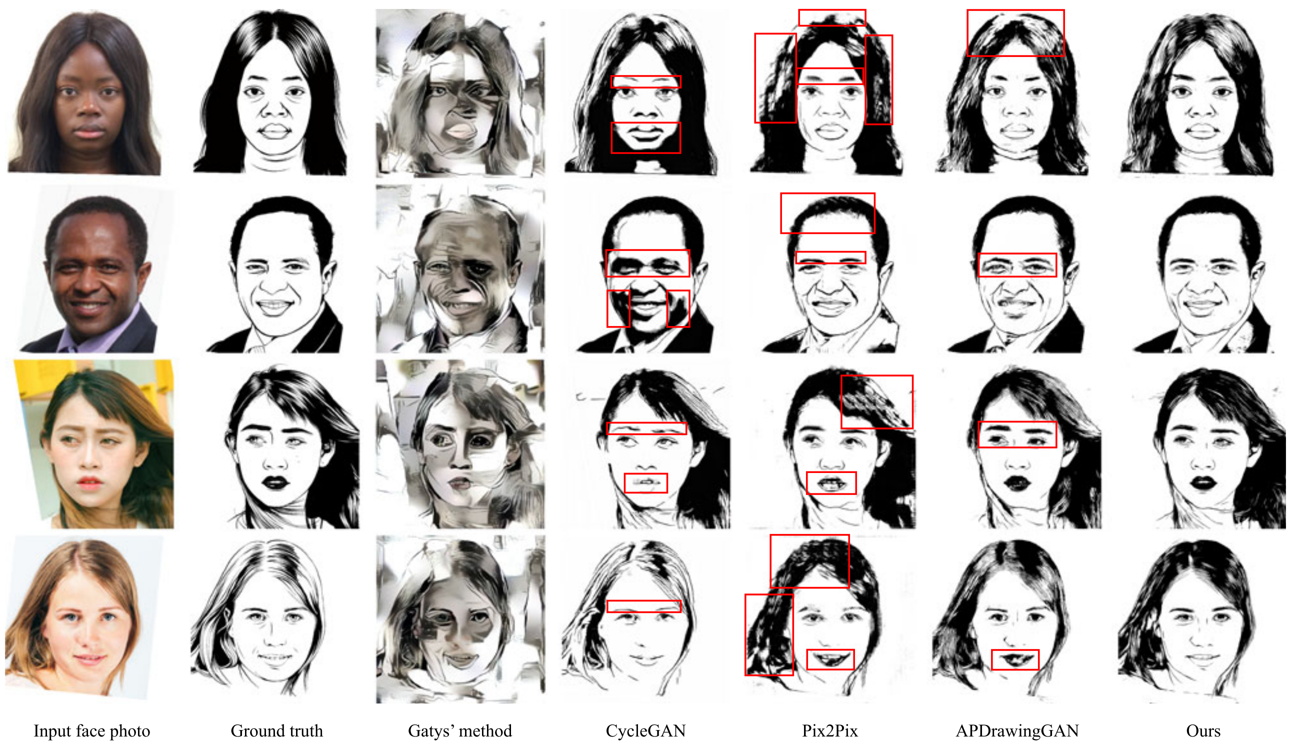


Fig. 9. Comparison results with Gatys' method [1], CycleGAN [3], Pix2Pix [2], APDrawingGAN [8] and APDrawingGAN++.

ensure fairness. Section 7.1 presents a detailed comparison with state of the arts (including both qualitative and quantitative evaluation). Section 7.2 presents a comprehensive ablation study in APDrawingGAN++.

7.1 Comparison With State-of-the-Art

7.1.1 Qualitative Evaluation

We compare APDrawingGAN++ with seven state-of-the-art style transfer methods: Gatys [1], CNNMRF [7], Deep Image Analogy [6], Pix2Pix [2], CycleGAN [3], Headshot Portrait [9] and APDrawingGAN [8].

Qualitative results of comparison with Gatys, CycleGAN, Pix2Pix and APDrawingGAN are shown in Fig. 9. Gatys' method [1] by default takes one content image and one style image as input. For fair comparison, we use all the style images in the training set and compute the average Gram matrix to model the target style as in [3]. As shown in Fig. 9, Gatys' method generates poor results for APDrawing stylization: some facial features are missing in the stylized results, and different regions are stylized inconsistently. The reasons behind these artifacts are that the method models style as texture information in the Gram matrix, which cannot capture our target style with little texture, and its content loss based on VGG output cannot preserve facial features precisely.

For CycleGAN, Pix2Pix and APDrawingGAN, we use the same training data as APDrawingGAN++ and default parameters to train the models. CycleGAN [3] cannot mimic the artistic portrait style well. As shown in Fig. 9, CycleGAN's results do not look like an artist's drawing, especially in the facial features. There are many artifacts, such as missing details in the eyes, blurred/dithered lip region, dark patches (e.g., the eyes and cheeks in the second row) caused by shadows, and not capturing eyebrow style. CycleGAN is unable to preserve facial features because it uses the cycle-consistency to constrain the content, which is less accurate than a supervised method and leads to problems when one of the domains is not accurately recovered. Also note that although CycleGAN uses our APDrawing dataset in an unpaired way, the images are actually paired: the actual paired training data makes it easier for CycleGAN to learn APDrawing style. Without paired data, the results will be worse.

Pix2Pix [2] generates results that preserve some aspects of artistic drawings, but they also have many artifacts. There are many messy unwanted lines, making the stylized result unlike the input photo, and the white lines in the hair are not learned well. The reason is that a generator with one CNN is unable to learn several independent drawing techniques in different facial regions, and there is no specifically designed loss term dedicated to the APDrawing style.

APDrawingGAN [8] generates drawings that capture different drawing techniques in different facial regions and have delicate white lines in the hair. However, as shown in Fig. 9, APDrawingGAN's results have less delicate facial features (e.g., the eyes of the second and third rows, the lip of the bottom row). Moreover, APDrawingGAN cannot handle dark faces well, leading to white patches in the hair or messy lines in the face (e.g., many white patches in the hair in the first row, many messy lines in the face in the second row). In comparison, our method generates high-quality results (i.e., with delicate facial features and white lines in the hair) and can adapt well with faces over a wide range of colors.

Qualitative results of comparison with CNNMRF, Deep Image Analogy and Headshot Portrait are shown in Fig. 10. These methods take one content image and one style image as input, and require these two images to be similar. Given a content image in the test set, we select two style images in the

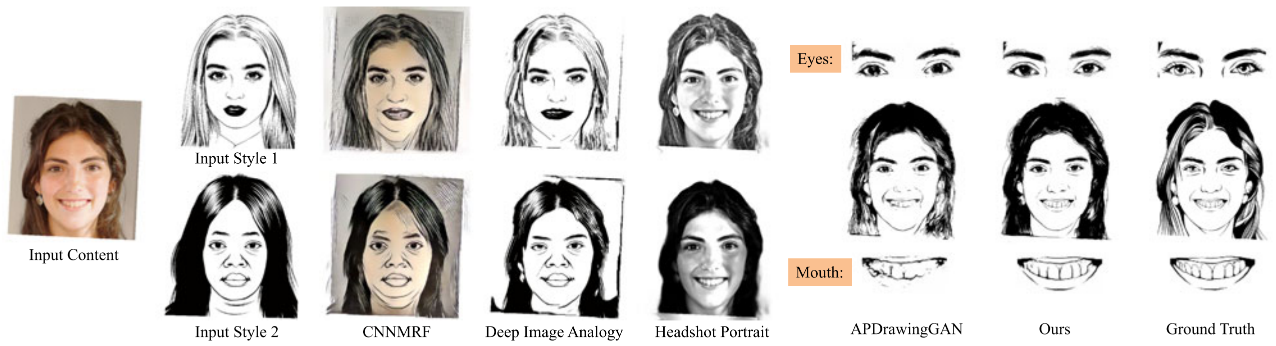


Fig. 10. Comparison results with CNNMRF [7], Deep Image Analogy [6], Headshot Portrait [9], APDrawingGAN [8], and APDrawingGAN++.

training set that are semantically similar to the content image (i.e., they have similar facial features) as shown in Fig. 10. Each pair of content and style images can generate one result. We draw the following observation from these two stylized results. CNNMRF [7] generates results that do not exhibit the same color distribution as the target style. Both CNNMRF and Deep Image Analogy [6] generate results with facial features closer to the style image but unlike the input content image, i.e., content has been erroneously copied from the style image. Headshot Portrait [9] is a portrait specific method but it generates photo-realistic results, which are not the style of the target artist's portrait drawing. In comparison, our method generates drawings that both preserve the facial features in the face photo and capture the artistic portrait drawing style. Moreover, our results are high-quality and are also better than APDrawingGAN with respect to the fine details, as highlighted by the close-ups of facial features.

In a nutshell, our method is significantly better than Gatys' method, CNNMRF, Deep Image Analogy and Headshot Portrait. Our method is also better than Pix2Pix, CycleGAN and APDrawingGAN in handling fine details. Below we present a quantitative evaluation that compares our method with the latter three methods.

7.1.2 Quantitative Evaluation

Human Preference. Due to the subjective nature of image styles, we conduct a user study with 102 participants to compare our APDrawingGAN++ results to CycleGAN, Pix2Pix, and APDrawingGAN. All 70 test pairs were used in the user study. For four stylized drawings in an image group, any two of them, i.e., (CycleGAN, Pix2Pix), (CycleGAN, APDrawingGAN), (CycleGAN, APDrawingGAN++), (Pix2Pix, APDrawingGAN), (Pix2Pix, APDrawingGAN++) and (APDrawingGAN, APDrawingGAN++), were compared once by each participant. For each of the four methods (CycleGAN, Pix2Pix, APDrawingGAN and APDrawingGAN++), we compute the percentage of it being ranked best, and the percentage of it being preferred in pairwise comparison. The results are summarized in Table 1. Results show that our APDrawingGAN++ is ranked the best in 67.48 percent of cases, significantly higher than CycleGAN and Pix2Pix where each of them is only ranked the best for less than 10 percent of cases, and much higher than APDrawingGAN which is ranked the best for 16.74 percent of cases. The percentage of our APDrawingGAN++ being preferred in pairwise comparison is 79.45 percent, much higher than CycleGAN, Pix2Pix and APDrawingGAN. We then conduct analysis of variance (ANOVA) for pairwise comparisons on ranked best percentage and preferred percentage. Pairwise ANOVA results are shown in Table 2. All of the p-values are $\ll 0.01$, justifying that the rejection of the null hypothesis and the differences between the means of our method and another method (Pix2Pix, CycleGAN or APDrawingGAN) are statistically significant. A test boxplot of four methods is shown in Fig. 12. We further investigate the improvement of APDrawingGAN++ over APDrawingGAN on dark faces. We concentrate on the pairwise comparison of (APDrawingGAN, APDrawingGAN++) and find that APDrawingGAN++ wins in this comparison

TABLE 1
Human Preference Statistics

Methods	Ranked Best	Preferred in a Pair
CycleGAN [3]	9.35%(2.6%∼14.7%)	31.87% (22.9%∼40.6%)
Pix2Pix [2]	6.43%(3.5%∼9.8%)	31.83% (27.7%∼35.1%)
APDrawingGAN [8]	16.74% (10.9%∼23.8%)	56.85% (53.2%∼59.6%)
APDrawingGAN++	67.48% (55.7%∼79.1%)	79.45% (70.0%∼87.5%)

For each of the four methods (CycleGAN, Pix2Pix, APDrawingGAN and APDrawingGAN++), the average percentage of it being ranked best, and the average percentage of it being preferred over another method are summarized. The 90 percent confidence intervals are shown in parentheses.

TABLE 2
Analysis of Variance (ANOVA) Results for Pairwise Comparisons

Pairwise comparison	Ranked Best	Preferred in a Pair
Ours vs CycleGAN	p=2.20e-17	p=1.07e-16
Ours vs Pix2Pix	p=6.48e-19	p=2.72e-19
Ours vs APDrawingGAN	p=4.90e-16	p=2.97e-12

TABLE 3
Average LPIPS Distance of CycleGAN, Pix2Pix, APDrawingGAN and Our APDrawingGAN++ on the Full Test Set

Methods	LPIPS
CycleGAN [3]	0.345
Pix2Pix [2]	0.330
APDrawingGAN [8]	0.291
APDrawingGAN++	0.258

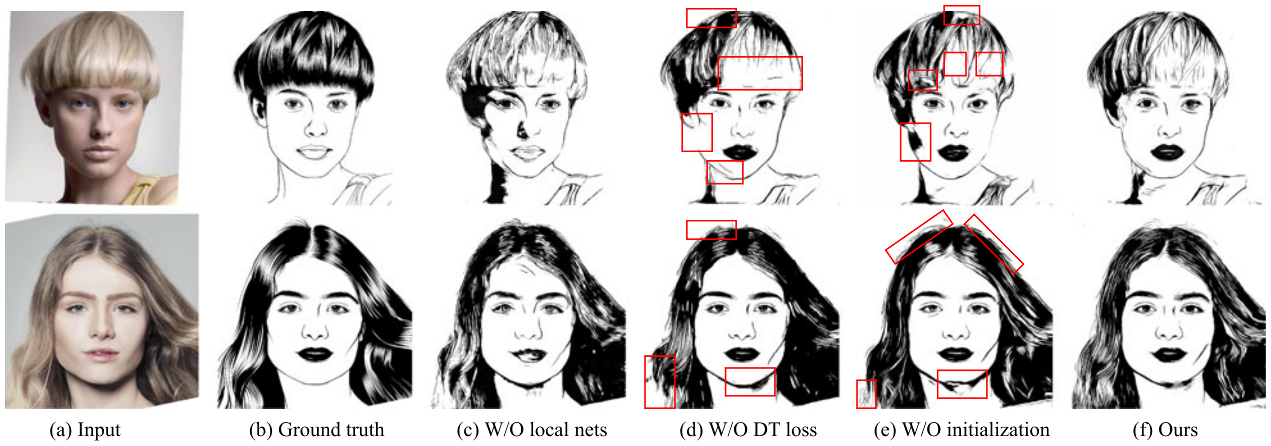


Fig. 11. Ablation study: (a) input face photos, (b) ground truth drawings by an artist, (c) results of removing local networks G_{l*}, D_{l*}, and E_* in APDrawingGAN++, (d) results of removing line-promoting DT loss L_{DT} from Eq.(1), (e) results of not using model pre-trained on NPR data as initialization, and (f) our results.

in 77.67 percent of cases for dark faces and 74.69 percent of cases for light faces. This justifies that histogram matching augmentation in APDrawingGAN++ is particularly useful for correcting dark faces that cannot be dealt with well by APDrawingGAN. The full details on how to set up the user study are presented in Section A5 of the appendix, available in the online supplemental material.

LPIPS Distance. We compare APDrawingGAN++ with CycleGAN, Pix2Pix and APDrawingGAN using the LPIPS distance [50], which is a perceptual similarity metric and has been demonstrated to correlate well with human perceptual similarity. We evaluate the LPIPS distance on the full test set to measure the perceptual similarity between generated APDrawings and real APDrawings using the authors' code[3] and default settings. The comparison results are presented in Table 3. The results show that our method has a much lower LPIPS distance than CycleGAN and Pix2Pix, and lower than APDrawingGAN, indicating our generated APDrawings are perceptually closer to the real APDrawings.

7.2 Ablation Study in APDrawingGAN++

We perform an ablation study from two aspects: one is on key ingredients in APDrawingGAN++ and the other is on the difference between APDrawingGAN++ and APDrawingGAN. More studies on other ingredients are presented in the Appendix, available in the online supplemental material.

7.2.1 *Study on Local Networks, Line-Promoting DT Loss, and Initialization*

First, we perform an ablation study on key ingredients in APDrawingGAN++, i.e., local networks G_{l*}, D_{l*} and E_*, line-promoting DT loss L_{DT} and initialization.

Local networks (G_{l*}, D_{l*} and E_*) in APDrawingGAN++ are essential to capture the style of each facial region. Since the style of an APDrawing contains several independent rendering techniques in different local regions, without local networks, the model cannot learn the varying styles well with a location-independent fully convolutional network. As shown in Fig. 11c, without local networks, the model generates messy results, where both facial region and hair region exhibit messy hairy style, leading to obvious defects.

Line-promoting DT loss L_{DT} is essential to produce good and clean results with delicate lines. Without the DT loss, there are fewer delicate lines in the hair region and some undesirable white patches appear instead, as shown in both rows in Fig. 11d. Moreover, some unattractive lines appear around the jaw, leading to drawings unlike the input photos, as shown in both results in Fig. 11d. These lines are effectively avoided by using the DT loss.

Initialization using the model pre-trained on the NPR data helps the model to generate good results in less time. The results without initialization are worse in having more messy lines in the facial region and more messy hair region, as shown in the cheek region of the first result, the chin region of the second result and the hair region of both

3. https://github.com/richzhang/PerceptualSimilarity. We use version 0.1.

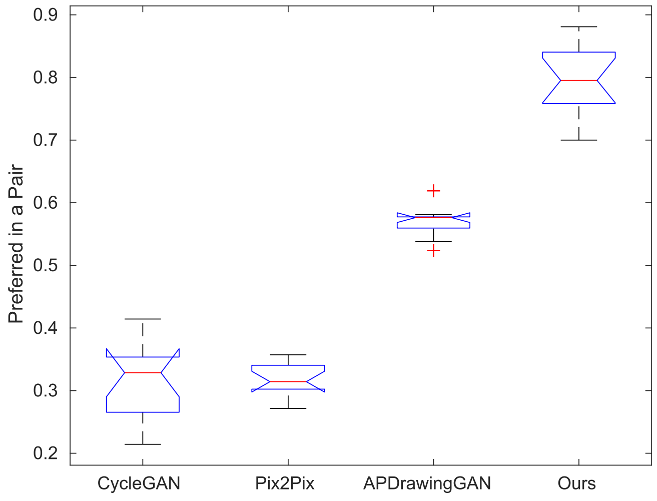


Fig. 12. Test boxplot [49] of four methods. In each box, the central red line indicates the median, and the bottom and top blue edges of the box indicate the 25 and 75 percent percentiles respectively. The dashed black line extends to those extreme data points which are not considered as outliers. The outliers (if any) are plotted using the red '+' symbol.

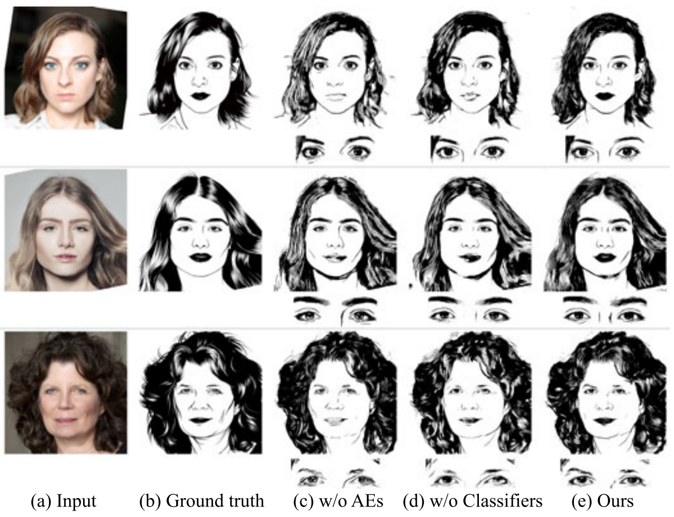


Fig. 13. Ablation study on auto-encoders and classifiers: (a) input face photos, (b) ground truth drawings by an artist, (c) results of removing auto-encoders in generator, (d) results of removing lip and hair classifiers, (e) our results. To better compare changes resulted by auto-encoders, zoomed eyes are shown under each synthesized drawing. Zoom in to see more facial feature details for lips and hair.

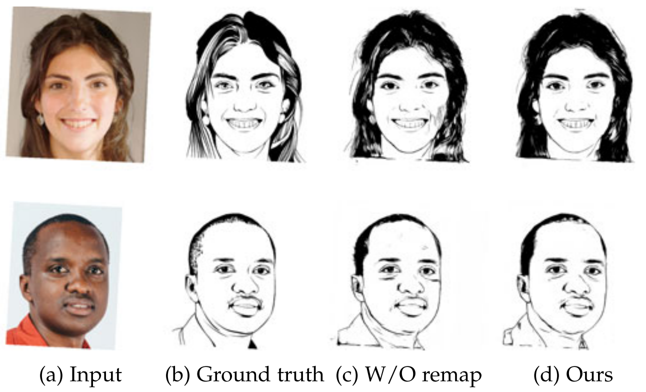


Fig. 15. Ablation study on DT loss nonlinear mapping: (a) input face photos, (b) ground truth drawings by an artist, (c) results of not using nonlinear mapping in DT loss, and (d) our results.

results in Fig. 11e. The pre-training strategy helps the model to quickly converge to a good result, avoiding such artifacts.

7.2.2 Study on Difference Between APDrawingGAN++ and APDrawingGAN

The major differences between APDrawingGAN++ and APDrawingGAN include (1) auto-encoders, (2) classifiers for lips and hair, (3) histogram matching augmentation, (4) DT nonlinear mapping in the DT loss calculation and (5) a novel line continuity loss. We perform an ablation study on their effect.

Auto-encoders help generate better facial feature drawings in fine detail. As illustrated in Fig. 13c, without auto-encoders, the drawings around eyes, noses and lips are worse than using auto-encoders, i.e., with more messy lines around these facial features and less natural shape (see the eyes, lip and nose of all three examples). Classifiers C_{lip} and C_{hair} help get better lips, hair drawings and reasonable lip colors according to the input photos. As illustrated in Fig. 13d, without classifiers and the one-hot vector to control lip and hair drawing generation, the colors (white/black) of lip drawings are often not consistent with the artist drawings (lips with lipstick are often drawn in full black by the artist, but are drawn in either full white or half white half black in Fig. 13d), and the hair drawings tend to have more unwanted white patches.

Histogram matching augmentation (Haug) helps balance face colors in the training set. Since there are relatively few dark faces in the APDrawing dataset, we found histogram matching augmentation especially helps portrait generation of dark faces. We use dark face photos to illustrate its improvement in Fig. 14. Without Haug, the synthesized drawings are more messy and may have unwanted black patches in the faces.

DT nonlinear mapping in the DT loss calculation helps strongly penalize large misalignments while tolerating small misalignments. As illustrated in Fig. 15, without DT mapping, the synthesized drawings are more messy, while with DT mapping, we get cleaner drawings.

Line continuity loss helps improve line quality in generated APDrawings. As illustrated in Fig. 16, without the line continuity loss, the lines in synthesized drawings are often discontinuous, while with line continuity loss, the lines in synthesized drawings become more continuous. Overall, using the line continuity loss lowers the average LPIPS distance on the test set from 0.277 to 0.258.

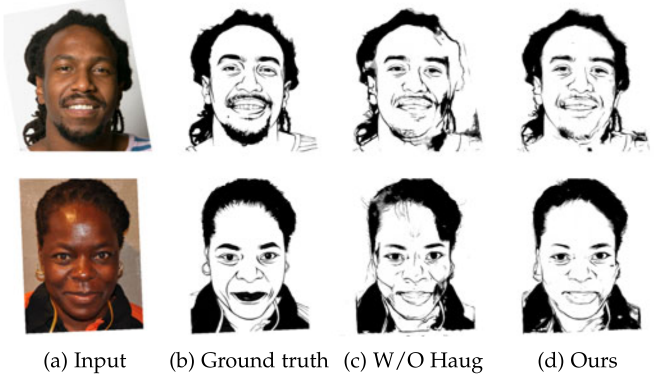


Fig. 14. Ablation study on histogram matching augmentation (Haug): (a) input face photos, (b) ground truth drawings by an artist, (c) results of not using histogram matching augmentation, and (d) our results.

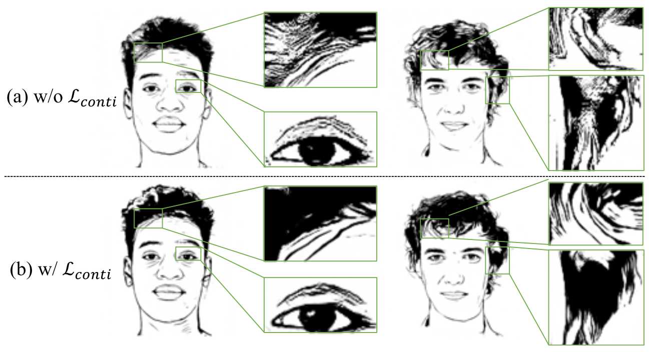


Fig. 16. Ablation study on line continuity loss: (a) synthesized drawings with line continuity loss removed (b) synthesized drawings with line continuity loss included. To better compare changes due to the line continuity loss, zoomed local regions are shown beside each synthesized drawing.

8 Conclusion

In this paper, we propose APDrawingGAN++, a composite GAN model to transform a face photo into a high-quality APDrawing. APDrawingGAN++ builds upon composite generators and discriminators that combine both a global network (for images as a whole) and local networks (for individual facial regions). We propose a loss function dedicated to APDrawing with five loss terms, including a novel DT loss (to promote line-stroke-based style in APDrawings), a novel line continuity loss (to enhance line continuity in APDrawings) and a local transfer loss (for local networks to preserve facial features).

APDrawingGAN++ is dedicated to the human face and APDrawing style, and particularly aims to avoid the many artifacts produced by existing methods. In particular, APDrawingGAN++ improves upon APDrawingGAN in the following six aspects: (1) auto-encoders are introduced to improve facial feature drawings; (2) two lip and hair classifiers are introduced to guide the local generator/auto-encoder towards a desired style; (3) a nonlinear mapping on the DT loss is used to strongly penalize large misalignments while tolerating small misalignments; (4) a line continuity loss is introduced to enhance line continuity in generated APDrawings; (5) a theoretical explanation of using multiple generators for APDrawing generation; and (6) histogram matching augmentation is used for the training set to achieve better results for dark faces.

Experimental results and a user study show that APDrawingGAN++ can generate high-quality and expressive APDrawings and outperforms state-of-the-art methods. In particular, APDrawingGAN++ generates APDrawings with improved facial features than APDrawingGAN and can adapt well with faces over a wide range of colors.

Acknowledgments

This work was supported by the Natural Science Foundation of China (61725204, 61521002) and Royal Society-Newton Advanced Fellowship (NA150431).

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2414–2423.

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

[3] 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.*, 2017, pp. 2223–2232.

[4] S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, and T. Darrell, "Multi-content GAN for few-shot font style transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7564–7573.

[5] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "CartoonGAN: Generative adversarial networks for photo cartoonization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9465–9474.

[6] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 120:1–120:15, 2017.

[7] C. Li and M. Wand, "Combining Markov random fields and convolutional neural networks for image synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2479–2486.

[8] R. Yi, Y.-J. Liu, Y.-K. Lai, and P. L. Rosin, "ApdrawingGAN: Generating artistic portrait drawings from face photos with hierarchical GANs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10 743–10 752.

[9] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand, "Style transfer for headshot portraits," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 148:1–148:14, 2014.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 262–270.

[11] J. Johnson, A. Alahi, and L. Fei-Fei , "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.

[12] J. E. Kyprianidis, J. P. Collomosse, T. Wang, and T. Isenberg, "State of the "Art": A taxonomy of artistic stylization techniques for images and video," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 5, pp. 866–885, May 2013.

[13] P. L. Rosin *et al.*, "Benchmarking non-photorealistic rendering of portraits," in *Proc. Int. Symp. Non-Photorealistic Animation Rendering*, 2017, pp. 11:1–11:12.

[14] M. Zhao and S. C. Zhu, "Portrait painting using active templates," in *Proc. Int. Symp. Non-Photorealistic Animation Rendering*, 2011, pp. 117–124.

[15] T. Wang, J. P. Collomosse, A. Hunter, and D. Greig, "Learnable stroke models for example-based portrait painting," in *Proc. Brit. Mach. Vis. Conf.*, 2013, pp. 36.1–36.11.

[16] I. Berger, A. Shamir, M. Mahler, E. Carter, and J. Hodgins, "Style and abstraction in portrait sketching," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 55:1–55:12, 2013.

[17] H. Chen, Z. Liu, C. Rose, Y. Xu, H. Shum, and D. Salesin, "Example-based composite sketching of human portraits," in *Proc. Int. Symp. Non-Photorealistic Animation Rendering*, 2004, pp. 95–153.

[18] M. Meng, M. Zhao, and S. C. Zhu, "Artistic paper-cut of human portraits," in *Proc. Int. Conf. Multimedia*, 2010, pp. 931–934.

[19] Y. Zhang *et al.*, "Data-driven synthesis of cartoon faces using different styles," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 464–478, Jan. 2017.

[20] P. L. Rosin and Y. Lai, "Non-photorealistic rendering of portraits," in *Proc. Workshop Comput. Aesthetics*, 2015, pp. 159–170.

[21] A. Selim, M. A. Elgharib, and L. Doyle, "Painting style transfer for head portraits using convolutional neural networks," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 129:1–129:18, 2016.

[22] J. Fišer *et al.*, "Example-based synthesis of stylized facial animations," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 155:1–155:11, 2017.

[23] P. L. Rosin and Y. Lai, "Towards artistic minimal rendering," in *Proc. Int. Symp. Non-Photorealistic Animation Rendering*, 2010, pp. 119–127.

[24] H. Winnemöller, J. E. Kyprianidis, and S. C. Olsen, "XDoG: An eXtended difference-of-Gaussians compendium including advanced image stylization," *Comput. Graph.*, vol. 36, no. 6, pp. 740–753, 2012.

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

[26] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

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

[28] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[29] X. Shen *et al.*, "Automatic portrait segmentation for image stylization," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 93–102, 2016.

[30] H. Chang, J. Lu, F. Yu, and A. Finkelstein, "PairedCycleGAN: Asymmetric style transfer for applying and removing makeup," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 40–48.

[31] Y. Li, C. Fang, A. Hertzmann, E. Shechtman, and M.-H Yang, "Im2Pencil: Controllable pencil illustration from photographs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1525–1534.

[32] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "OpenFace: A general-purpose face recognition library with mobile applications," Carnegie Mellon University, Pittsburgh, PA, *Tech. Rep. CMU-CS-16–118*, 2016.

[33] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception GAN for photorealistic and identity preserving frontal view synthesis," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2439–2448.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[35] D. S. Ma, J. Correll, and B. Wittenbrink, “The Chicago face database: A free stimulus set of faces and norming data,” *Behav. Res. Methods*, vol. 47, no. 4, pp. 1122–1135, 2015.

[36] M. Walker, S. Schönborn, R. Greifeneder, and T. Vetter, “The Basel face database: A validated set of photographs reflecting systematic differences in Big Two and Big Five personality dimensions,” *PLoS One*, vol. 13, no. 3, 2018, Art. no. e0193190.

[37] R. Courset *et al.*, “The caucasian and north african french faces (CaNAFF): A face database,” *Int. Rev. Soc. Psychol.*, vol. 31, no. 1, pp. 22:1–22:10, 2018.

[38] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss, “The FERET database and evaluation procedure for face-recognition algorithms,” *Image Vis. Comput.*, vol. 16, no. 5, pp. 295–306, 1998.

[39] P. Peer, Ž. Emeršič, J. Bule, J. Žganec-Gros, and V. Štruc, “Strategies for exploiting independent cloud implementations of biometric experts in multibiometric scenarios,” *Math. Problems Eng.*, vol. 2014, pp. 1–15, 2014.

[40] N. C. Ebner, M. Riediger, and U. Lindenberger, “FACES-A database of facial expressions in young, middle-aged, and older women and men: Development and validation,” *Behav. Res. Methods*, vol. 42, no. 1, pp. 351–362, 2010.

[41] C. E. Thomaz and G. A. Giraldi, “A new ranking method for principal components analysis and its application to face image analysis,” *Image Vis. Comput.*, vol. 28, no. 6, pp. 902–913, 2010.

[42] N. Strohminger, K. Gray, V. Chituc, J. Heffner, C. Schein, and T. B. Heagins, “The MR2: A multi-racial, mega-resolution database of facial stimuli,” *Behav. Res. Methods*, vol. 48, no. 3, pp. 1197–1204, 2016.

[43] O. Chelnokova *et al.*, “Rewards of beauty: The opioid system mediates social motivation in humans,” *Mol. Psychiatry*, vol. 19, pp. 746–751, 2014.

[44] T. F. Vieira, A. Bottino, A. Laurentini, and M. De Simone, “Detecting siblings in image pairs,” *The Vis. Comput.*, vol. 30, no. 12, pp. 1333–1345, 2014.

[45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[46] S. M. Pizer *et al.*, “Adaptive histogram equalization and its variations,” *Comput. Vis. Graph. Image Process.*, vol. 39, no. 3, pp. 355–368, 1987.

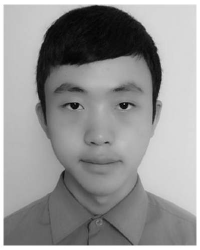
[47] E. Simo-Serra , S. Iizuka, and H. Ishikawa, “Mastering sketching: Adversarial augmentation for structured prediction,” *ACM Trans. Graph.*, vol. 37, no. 1, pp. 11:1–11:13, 2018.

[48] A. Paszke *et al.*, “Automatic differentiation in PyTorch,” in *Proc. 31st Conf. Neural Inf. Process. Syst. Workshop*, 2017, pp. 1–4.

[49] R. V. Hogg and J. Ledolter, *Engineering Statistics*. New York, NY, USA: MacMillan, 1987.

[50] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 586–595.

Ran Yi received the BEng degree from Tsinghua University, China, in 2016, where she is currently working toward the PhD degree in the Department of Computer Science and Technology. Her research interests include computational geometry, computer vision, and computer graphics.



Mengfei Xia is currently working toward the undergraduate degree in the Department of Mathematical Science, Tsinghua University, China. He won the silver medal twice in 30th and 31st National Mathematical Olympiad of China. His research interests include mathematical foundation in deep learning, image processing, and computer vision.

Yong-Jin Liu (Senior Member, IEEE) received the BEng degree from Tianjin University, China, in 1998, and the PhD degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 2004. He is currently a professor with the Department of Computer Science and Technology, Tsinghua University, China. His research interests include computational geometry, computer graphics and computer vision. For more information, please visit https://cg.cs.tsinghua.edu.cn/people/∼Yongjin/Yongjin.htm

Yu-Kun Lai (Member, IEEE) received the BS and PhD degrees in computer science from Tsinghua University, China, in 2003 and 2008, respectively. He is currently a reader at the School of Computer Science and Informatics, Cardiff University, UK. His research interests include computer graphics, computer vision, geometric modeling and image processing. For more information, please visit https://users.cs.cf.ac.uk/Yukun.Lai/

Paul L. Rosin (Member, IEEE) is currently a professor at the School of Computer Science and Informatics, Cardiff University, UK. He was a lecturer with the Department of Information Systems and Computing, Brunel University London, UK, research scientist with the Institute for Remote Sensing Applications, Joint Research Centre, Ispra, Italy, and UK Curtin University of Technology, Perth, Australia. His research interests include low level image processing, facial analysis, non-photorealistic rendering, and cultural heritage. For more information, please visit http://users.cs.cf.ac.uk/Paul.Rosin/