Scribbler: Controlling Deep Image Synthesis with Sketch and Color

Patsorn Sangkloy[1],   Jingwan Lu[2],   Chen Fang[2],   Fisher Yu[3],   James Hays[1]

[1]Georgia Institute of Technology   [2]Adobe Research   [3]Princeton University

Generated results

Figure 1. A user can sketch and scribble colors to control deep image synthesis. On the left is an image generated from a hand drawn sketch. On the right several objects have been deleted from the sketch, a vase has been added, and the color of various scene elements has been constrained by sparse color strokes. For best resolution and additional results, see scribbler.eye.gatech.edu

Abstract

Several recent works have used deep convolutional networks to generate realistic imagery. These methods sidestep the traditional computer graphics rendering pipeline and instead generate imagery at the pixel level by learning from large collections of photos (e.g. faces or bedrooms). However, these methods are of limited utility because it is difficult for a user to control what the network produces. In this paper, we propose a deep adversarial image synthesis architecture that is conditioned on sketched boundaries and sparse color strokes to generate realistic cars, bedrooms, or faces. We demonstrate a sketch based image synthesis system which allows users to scribble over the sketch to indicate preferred color for objects. Our network can then generate convincing images that satisfy both the color and the sketch constraints of user. The network is feed-forward which allows users to see the effect of their edits in real time. We compare to recent work on sketch to image synthesis and show that our approach generates more realistic, diverse, and controllable outputs. The architecture is also effective at user-guided colorization of grayscale images.

1. Introduction

Recently, numerous image synthesis methods built on neural networks have emerged [41, 25, 12, 37, 21, 13]. These methods can generate detailed and diverse (if not quite photorealistic) images in many domains. However, it is still unclear how to *control* these powerful new tools. How can we enable everyday users (non-artists) to harness the power of deep image synthesis methods and produce realistic imagery? Several recent methods have explored controllable deep synthesis [8, 50, 55, 14, 56, 18, 46] and we focus on two complementary forms of control – sketches and

color strokes. Sketches are a compelling form of control because anyone can draw (potentially very badly) and because it is easy to edit sketches, e.g. to remove or add objects, whereas the equivalent operations in the image domain require artistic expertise. Color is a compelling form of control because many sketches or grayscale scenes are fundamentally ambiguous with respect to color [54], but it is easy for a user to intervene, e.g. to scribble that drapes should be blue and the valance should be red (Figure 1). Both forms of control are relatively sparse and require a deep network to synthesize image detail beyond what is contained in the input. The deep network must also implicitly learn a significant amount of high-level image understanding, e.g. what colors are allowable for particular objects, the boundaries of objects such that color does not bleed beyond a single semantic region, and the appropriate high frequency textures for different scene elements.

We propose a deep adversarial (GAN) image synthesis architecture trained to generate realistic images from sparse and simple sketched boundaries and color strokes. We train our network on a diverse set of synthetic sketches optionally augmented with randomly sampled color strokes. The network learns to recover the color and detail lost to the sketching process and to extrapolate the sparse color indications to semantic scene elements. We show qualitative results of image synthesis in three domains – faces, cars, and bedrooms. We test on synthetic sketches as well as imperfect hand-drawn sketches.

Our approach is similar to Sketch Inversion [14], which also generates images from sketches, although we show the benefit of adversarial training, introduce color control signals, demonstrate results on image domains beyond faces, and demonstrate that users can perform simple edits to sketches to control the synthesis. Our control signals are

most similar to Zhu et al. [56] – they also demonstrate that GANs can be constrained by sketch and color strokes. However, our architecture is a feed-forward mapping from sketch and color to images while Zhu et al. perform an optimization to map user sketches into the latent GAN space in order to find the most similar image on the natural image manifold (as understood by the GAN). Their approach does not see user inputs at training time and thus cannot learn the complex mapping between user inputs and desired image outputs. Their method is also significantly slower because it is not a strictly feed-forward process and this hinders interactive image editing. The concurrent work of Isola et al. [18] significantly overlaps with our own. Both approaches use conditional GANs for the sketch to photo as well as grayscale to color synthesis tasks, although they do not focus on user control of the synthesis.

The contributions of this paper include:

- We are the first to demonstrate an adversarial deep architecture that can learn to generate realistic images from imperfect sketches with sparse color 'scribbles'. Our feed-forward architecture is fast and interactive.

- We improve the quality of sketch-to-image synthesis compared to existing work [14]. We produce higher resolution, more diverse images spanning more image domains (bedrooms and cars in addition to faces).

- Our method can generate realistic images from diverse sketch styles, including imperfect human sketches or synthetic sketches. We achieve this generality by augmenting our training data with multiple sketch styles.

- Finally, we demonstrate that our adversarial architecture is also promising for image colorization. We show encouraging results for grayscale to RGB conversion and introduce controllable colorization using the same sparse color strokes used with sketches.

2. Related Work

Synthesizing images by learning from image collections is a long standing interest of the computer graphics and vision communities. Previously, the most successful methods tended to be non-parametric approaches which found clever ways to reuse existing image fragments [30, 9, 15, 22, 6, 3].

In the last few years, *parametric* models built on deep convolutional networks have shown promising results [12, 8, 37, 21, 13]. While deep image synthesis methods cannot yet create realistic, high-resolution images they have an implicit ability to generalize that is difficult for data-driven non-parametric methods (e.g. the ability to hallucinate unseen viewpoints of particular chairs based on the appearance changes of other chairs [8]). Because our visual world is both enormously complex (with appearance depending on viewpoints, materials, attributes, object identity, lighting, etc.) *and* heavy-tailed, non-parametric methods are limited even in the "big data" era. But deep image synthesis methods might implicitly factorize our visual world and thus generalize to situations beyond the training examples.

A common approach to deep image synthesis is to learn a low dimensional latent representation that can later be used to reconstruct an image, e.g. with Variational Autoencoders (VAEs) [21] or Generative Adversarial Networks (GANs) [12]. In general, deep image synthesis can be conditioned on any input vector [46], such as attributes [50], 3d viewpoint parameters and object identity [8], image and desired viewpoint [55], or grayscale image [54, 17, 23] .

Generative Adversarial Networks (GANs) Among the most promising deep image synthesis techniques are Generative Adversarial Networks (GANs) [12, 37] in which a *generative* network attempts to fool a simultaneously trained *discriminator* network that classifies images as real or synthetic. The discriminator discourages the generator from producing obviously fake images. In particular, straightforward regression loss for image synthesis often leads to 'conservative' networks which produce blurry and desaturated outputs close to the mean of the data yet perceptually unrealistic. After training, the generator network is able to produce diverse images from a low dimensional latent input space. Although optimizing in this latent space can be used to 'walk' the natural image manifold (e.g. for image editing [4, 56] or network visualization [31, 32]), the space itself is not semantically well organized – the particular dimensions of the latent vector do not correspond to semantic attributes although mapping them to an intermediate structure image [47] can give us more insight.

Conditional GANs Instead of synthesizing images from latent vectors, several works explore *conditional* GANs where the generator is conditioned on more meaningful inputs such as text [39, 38], low resolution images (super-resolution) [24, 20], or incomplete images (inpainting) [35, 34, 51]. Conditional GANs have also been used to transform images into different domains such as product images [52] or different artistic styles [27]. One can also condition the *discriminator* on particular inputs, e.g. Reed et al. [39] condition both the generator and discriminator on an embedding of input text, which makes the discriminator more powerful. In this paper, only our generator is conditioned on input sketches and color strokes leaving the discriminator to discern real vs fake and not to evaluate the appropriateness of an output given the particular input.

Controlling deep image synthesis Several recent works share our motivation of adding user editable *control* to deep image generation. Examples of control signals include 3d pose of objects [8], natural language [39], semantic attributes [50], semantic segmentation [5], and object key-

points and bounding box [38].

The artistic style transfer approach of Gatys et al. [11] could also be considered a mechanism to control deep image synthesis. Their method does not 'learn' transformations end-to-end but instead uses a pre-trained network and optimizes for output images which have deep network feature *activations* (content) similar to one input image and deep network feature *correlations* (style) similar to another input image. The approach does not perform well for transformations which requires the synthesis of realistic detail (e.g. trying to preserve the 'content' of a sketch and the 'style' of a photograph).

The most similar controllable deep image synthesis approach is Zhu et al. [56] which optimizes for an image that is similar to an input sketch (potentially with color strokes) that lies on a learned natural image manifold. However, identifying a matching image within this manifold that is similar content-wise to the sketch can be challenging when the sketch and the image are significantly different. For the 'sketch brush' in [56], they get around this by optimizing for image with the same edges as user sketch that also lies within a natural image manifold as approximated by a pre-trained GAN. However, image edges are not necessarily a good proxy for human sketched strokes [42] and their method has no capacity to *learn* the mapping between user inputs and desired outputs. In contrast, our method enables control via sketch and color strokes in a unified framework learned end-to-end. Sketch Inversion [14] is also closely related to our work but they do not address color control. We compare our sketch-to-photo results with Sketch Inversion.

Controllable Colorization Our color control strokes are inspired by Colorization Using Optimization [26] which interpolates sparse color strokes such that color changes tend to happen at intensity boundaries. The algorithm does not learn the association between objects and colors and thus can only interpolate user provided colors (e.g. a tree in the background of a scene will not be green if the user only marked foreground objects). The algorithm also does not learn the spatial extent of objects, and thus colors might 'snap' to spurious boundaries or bleed over weak intensity edges that are none-the-less salient boundaries. Our deep network learns object color tendencies and object extent and thus can cleanly color objects either with no color strokes or with color strokes on a subset of scene elements (Figure 1). Similar control strokes have been applied to sketch and manga imagery [36, 45], but the results remain non-photorealistic and lack lighting and shading.

We are unaware of sparse scribbles being used as input constraints to deep generative networks, although Scribble-Sup [28] uses sparse scribbles to *supervise* the output of semantic segmentation networks. The scribbles are training data and there is no user control at test time.

Concurrent work Concurrent to our work, the 'pix2pix' method of Isola et al. [18] also uses conditional GANs for sketch to photo and grayscale to color synthesis. Additionally, they explore several other interesting image-to-image 'translation' tasks. Unlike our approach, they use a "U-Net" architecture [40] which allows later layers of the network to be conditioned on early layers where more spatial information is preserved. They condition both their generator and discriminator on the input whereas we condition only the generator. Their results are high quality and they are able to synthesize shoes and handbags from coarse sketches [10] even though their training data was simple image edges. In contrast, we take care to train on a diversity of synthetic sketch styles. The most significant difference between our works is that we introduce sparse color control strokes and demonstrate how to train a network so that it learns to intelligently interpolate such control signals, whereas Isola et al. [18] does not emphasize controllable synthesis.

3. Overview

In this paper, we explore adding direct and fine-grained user controls to generative neural networks. We propose a generic feed-forward network that can be trained end-to-end to directly transform users' control signals, for example a hand-drawn sketch and color strokes, to a high-res photo with realistic textural details.

Our proposed network is essentially a deep generative model that is conditioned on control signals. The network learns a transformation from control signal to the pixel domain. It learns to fill in missing details and colors in a realistic way. Section 3.1 discusses the network structure that is shared by all applications presented in the paper. Section 3.2 introduces the objective functions, in particular the combination of content loss and adversarial loss, which encourages the result to be photo-realistic while satisfying user's fine-grained control. Section 4 and 5 show how to enforce two different user controls in the proposed framework – using hand-drawn sketches to determine the gist or shape of the contents and using sparse color strokes to propagate colors to semantic regions. Section 6 applies the proposed framework in several interactive applications.

3.1. Network Architecture

We design a feed-forward neural network that takes an image as input and generates a photo of the same resolution as output. When generating an image conditioned on a high dimensional input in the same domain (i.e. from image to image), typically an encoder-decoder type of network architecture is adopted, for example in sketch inversion [14], image colorization [54, 17], and sketch simplification [43]. In a typical network structure, the input gets downsampled several times to a lower dimension, then goes through a sequence of non-linear transformations, and finally gets up-
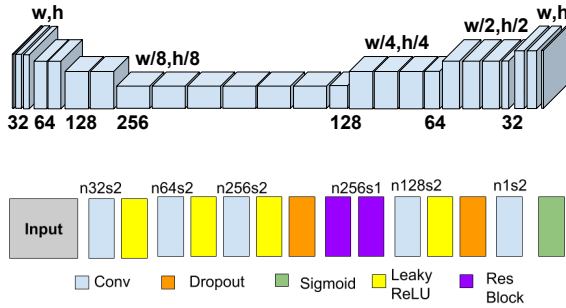
Figure 2. Network Architecture. For the generator (top), we follow the encoder-decoder design, and use three downsampling steps, seven residual blocks at the bottleneck resolution and three upsampling steps. Residual blocks use stride 1. Downsampling uses convolutions with stride 2. Upsampling uses bilinear upsampling followed by residual blocks. We use fully convolutional network for the discriminator (bottom). See scribbler.eye.gatech.edu for code and architecture details.

sampled to the desired output size. Recently, He et al. [16] proposed the residual connection that uses skip layers allowing network blocks to learn only the *residual* component. The use of residual block eases the training of deeper networks which improves the capability of neural network for more complex tasks.

We employ an encoder-decoder architecture with residual connections. Starting from the network design in Sketch Inversion [14], we introduce several important modifications to improve the visual quality of output and accommodate higher resolution input and more challenging image categories, such as car and bedroom. In particular, we add one more up/downsampling layer and double the number of filters in all convolutional layers between the last downsampling layer and the first upsampling step. In addition, we replace the deconvolutional layers with the bilinear upsampling step followed by two residual blocks, due to the recent finding that deconvolutional layers have the tendency to produce checkerboard artifacts commonly seen in deep generative models [33]. Overall, our architecture has around 7.8 millions learnable parameters, while the Sketch Inversion network we implemented has around 1.7 millions. See Figure 2 for a diagram of our architecture.

3.2. Objective Function

Given pairs of training images (input, ground-truth), where the input image is derived from the ground-truth photo (synthetically generated sketches and color strokes in our case), the simplest and most common loss is the average per-pixel L2 difference between the generated image and the ground-truth, which we denote as $\mathbf{L_p}$.

Previous work [14] showed that adding a *feature* loss to the objective function is beneficial for image generation tasks. Feature loss $\mathbf{L_f}$ is defined as the L2 difference in a feature space, where a feature is extracted from a cer-

tain layer of a pre-trained neural network representing high-level information of images.

While pixel and feature losses are widely used to explicitly correlate synthesized output with input, using them alone is often not sufficient to generate diverse, realistic images. More importantly, in our problem setup, conditioning on coarse user controls leaves us with a highly ill-posed problem where the potential solution space is multimodal. Therefore, with only pixel and feature losses, the network tends to average over all plausible solutions, due to the lack of a loss which pushes for realism and diversity.

For image categories like face, the results tend to have similar skin tones [14]. For more complicated categories like cars and bedrooms, where the foreground and background contents can have large variety of shapes and colors, the results might not be visually plausible, since neutral colors are chosen by the network to minimize MSE. The second and third rows in Figure 3 demonstrate the problems.

To encourage more variations and vividness in results, we experiment with adding an adversarial loss to the objective function. Generative adversarial networks (GAN), proposed by Goodfellow et al [12], have attracted considerable attention recently. A generative network $\mathbf{G}_\theta$ is jointly trained with a discriminative adversarial network $\mathbf{D}_\phi$, so that the discriminator tries to distinguish between the generated images and ground-truth images, while the generator tries to fool the discriminator into thinking the generated result is real. Dosovitskiy et al [7] showed that complimenting the feature loss with an adversarial loss leads to more realistic results. The adversarial loss $\mathbf{L_{adv}}$ is defined as:

$$\mathbf{L_{adv}} = -\sum log\mathbf{D}_\phi(\mathbf{G}_\theta(x_i)) \qquad (1)$$

We find that adversarial loss is also beneficial for our sketch-based image synthesis problem (Figure 3). With adversarial training, the network puts less emphasis on exactly reproducing ground-truth, but instead focuses on generating more realistic results with plausible color and shape deviation from ground-truth.

Adversarial training tends to be unstable, especially at the start of training when the generator does not produce anything meaningful and the discriminator can easily distinguish between real and fake. We find that using a weak discriminator $\mathbf{D}_\phi$ helps stabilize the training. We also avoided conditioning the discriminator on the input image, as this tends to increase the instability [35]. In particular, we use a fully convolutional structure without fully connected layers and batch normalization. Section 7 introduces additional tricks for successful adversarial training.

Finally, we also add a total variation loss $\mathbf{L_{tv}}$ to encourage smoothness in the output [19].

Our final objective function becomes:

$$\mathbf{L} = \mathbf{w_p}\mathbf{L_p} + \mathbf{w_f}\mathbf{L_f} + \mathbf{w_{adv}}\mathbf{L_{adv}} + \mathbf{w_{tv}}\mathbf{L_{tv}} \qquad (2)$$
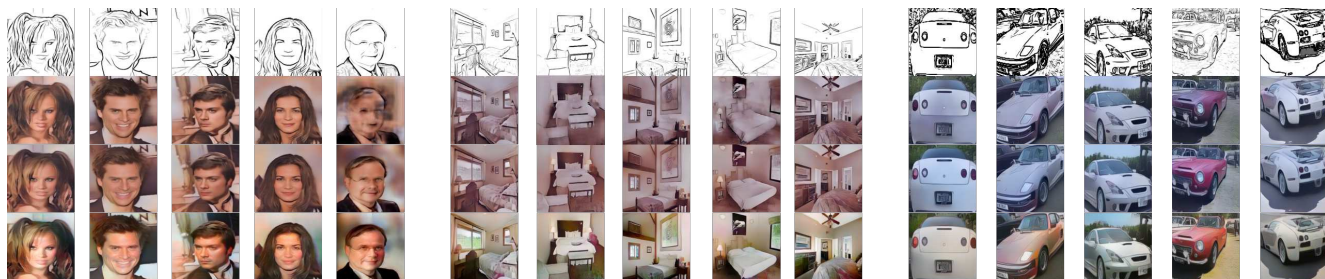
Figure 3. Results comparison. From top to bottom: input sketch, Sketch Inversion with content loss, our network with content loss, our network with content loss and adversarial loss

4. Sketch-based Photo Synthesis

In this section, we explore how to apply the proposed feed-forward network to hallucinate content, color and texture to reconstruct a photo based on an input sketch of arbitrary style. To train such a deep neural network, we need lots of training sketch-photo pairs. Though high quality hand-drawn sketches are readily available online, the corresponding photos based on which sketches are drawn are not. Therefore, we apply high-quality line drawing synthesis algorithms to generate synthetic sketches from photos. In order to handle real hand-drawn sketches at test time, we apply various data augmentations to the training data to improve the generality of the network. In this paper, we experiment with three image classes – faces [29], cars, and bedrooms [53]. We believe the proposed framework can generalize well to other categories given similar amounts of training data and training time.

4.1. Generation of Training Sketches

For each image category – face, car, or bedroom – we apply the boundary detection filter XDoG [49] on 200k photos to generate the corresponding synthetic sketches. The input (and output) resolution during training is 128x128.

To make the network invariant to the exact locations of the objects, we randomly crop both the input and the ground-truth images. For the face and bedroom categories, we first resize the images to 256x256 before randomly cropping them to 128x128. For the car category, we scale the images to 170x170 before cropping, since most cars already occupy large image areas, enlarging them too much means

Figure 4. We generate synthetic sketches from photos using five different algorithms. We also include and augment a small set of hand-drawn sketch-photo pairs to help generalize the network to handle real hand-drawn sketch inputs.

losing the global spatial arrangement and context.

In addition to random cropping, we also randomly adjust the brightness level of the sketch to get different levels of details from the same sketch (i.e. some sketch lines will disappear with higher brightness level). Finally, we also randomly cut off some lines in the sketch, by overlaying a random number of white strokes (the background color of sketch input) on top of the sketch. We randomize the length, width and locations of the white strokes.

4.2. Network Generalization

Real hand-drawn sketches exhibit a large variety of styles, from abstract pen-and-ink illustrations to elaborate pencil-like drawings with shading. The characteristics of the hand-drawn sketches might be very different from the synthetic sketches we generated algorithmically. Even with the various augmentations, random cropping, random brightness adjustment and random cut-off, the trained network might still overfit to that particular style of sketches. To improve the network generality, we further augment the training data by adding multiple styles of sketches.

For the face category, we obtain 20k additional images and for each image we randomly choose one of the following four algorithms to synthesize a corresponding sketch. See example sketches in Figure 4.

- **StyleNet** [11] We apply neural network-based style transfer algorithm to transfer the texture style of a pencil drawing to the ground-truth photo.

- **Photoshop filters** [2] Applying Photoshop's 'photocopy' effect to ground-truth images, we can generate two different versions of sketches with different levels of details and stroke darkness.

- **Gaussian blur on inverse image** [1] Using Photoshop, we can also synthesize another sketch style by performing Gaussian blur on an inverse (grayscale) image in Photoshop color dodge mode. This creates detailed line drawings with very little shading.

- **CUHK** Finally, we add the CUHK dataset, which contains 188 hand-drawn portrait sketches and their cor-
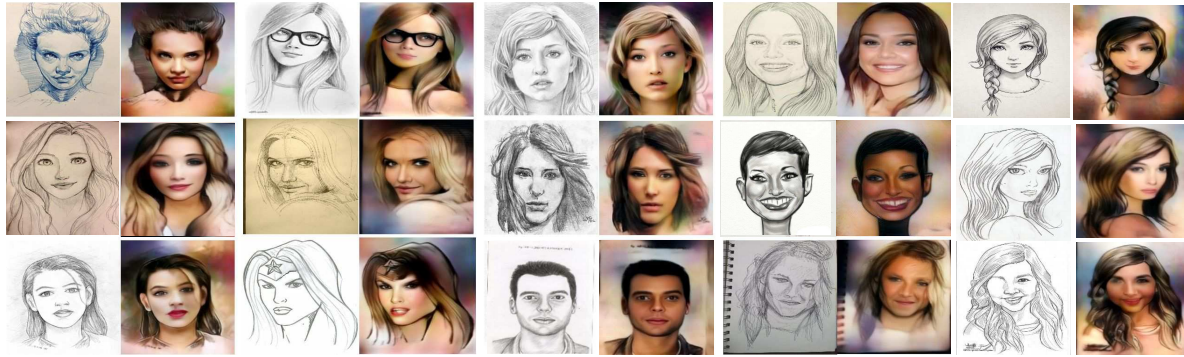
Figure 5. Sketch-based photo synthesis of hand-drawn test sketches. Despite the diverse sketch styles, our network usually produces high quality, diverse results. Notice the skin tone and hair color variations in the output. Some outputs are non-photorealistic because they are being somewhat faithful to caricatured input sketches. Unfortunately, some results have unrealistically high low-frequency contrast and appear unnaturally lit.

responding photos [48]. To give higher weights to the high quality hand-drawn sketches, we apply mirroring and varying degrees of rotation to the sketches and end up with 1869 images in total.

At this point, we have 21848 images of 6 different sketch styles. Pre-trained on the 200k sketches of the XDoG style, the network is fine-tuned using the 20k multi-style sketches. We use the same parameter settings as before and train the network on these additional data for 5 epochs.

4.3. Results and Discussions

For comparison purposes, we implemented the Sketch Inversion architecture as described in [14]. We trained both the Sketch Inversion network and our deeper network using the same training data and parameter settings. Figure 3 shows side-by-side comparisons of the results generated by Sketch Inversion (second row), our deeper network trained without (third row) and with adversarial loss (fourth row) on three different image categories. Compared to Sketch Inversion, our deeper network even without adversarial loss produces sharper results on complex bedroom scenes and performs better at hallucinating missing details (shapes of eyes and eyebrows) given simplified sketches with few lines. With adversarial loss, our network is encouraged to generate images with sharper edges, higher contrast and more realistic color and lighting. As discussed in Section 3.2, adversarial loss helps the network generate more diversified results, avoiding always producing similar skin tones and hair colors for portraits and dull and unrealistic colors for the bedrooms and cars. Figure 5 shows diverse hair colors and skin tones in the result.

Among the three image categories, bedroom is arguably most challenging, since each bedroom scene can contain multiple object categories. The fact that our current network handles it successfully with only 200K training data leads us to believe the possibility of training a general sketch-to-photo network across several image categories using an even deeper network.

After training with multiple sketch styles and various data augmentations (Section 4.2), our network generates much more realistic results given arbitrary hand-drawn sketches as input. Figure 5 shows reconstruction results based on sketches found by Google search. Note that the sketches are drawn with diverse styles, some detailed and realistic, some abstract and simplified. The results show that our network generalizes well to arbitrary hand-drawn sketches and is robust to the variations in head pose, background colors, and textures. Data augmentation such as random cropping and cutoff also helps our network hallucinate missing details. Figure 5 (bottom right) shows that the network can fill in the missing eye to some extent. However, generating missing object parts is a challenge itself, therefore we consider it beyond the scope of this paper.

The network trained with adversarial loss has an interesting behavior. When applying it to cartoonish or unprofessional sketches with exaggerated facial features, the network tends to 'realistify' or beautify the input sketch to generate result more photo-like at the cost of not strictly following the sketch constraints. For example, eyes that are inhumanly large will get reduced to a realistic size or faces with weird shapes will be smoothed and 'beautified' (see Figure 5). To produce realistic results, the network learns not to blindly trust the sketch input and instead uses its understanding of the natural image manifold acquired during the adversarial training.

5. User-guided Colorization

The previous section focuses on using gray-scale sketches to guide the generation of color photos. The lack of color information in the input causes the problem to be under-determined, since one sketch can correspond to photos colored in many different ways. Although the use of
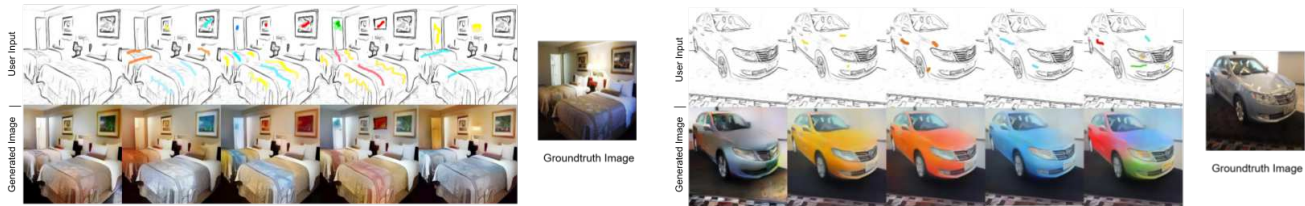
Figure 6. Guided sketch colorization results on held out test sketches for bedroom (left) and car (right). Without color strokes, our network produces results that closely follow the input sketch (leftmost column). With color strokes, our network adapts the results to satisfy different color constraints. Note that the fourth and fifth bedroom sketches are edited to add a framed picture on the wall and a lamp using only simple user edits.

adversarial loss constrains the output to lie on an approximated manifold of natural images and therefore limits the color choices, it is still up to the generator to choose a specific color.

In this section, we explore how to allow users to directly control the colors in the output. We modify the input to the network to include rough color information during training (Section 5.1). We investigate adding color controls in two applications: guided sketch colorization (Section 5.2) and guided image colorization (Section 5.3).

5.1. Generation of Training Color Strokes

One of the most intuitive ways to control the outcome of colorization is to 'scribble' some color strokes to indicate the preferred color in a region. To train a network to recognize these control signals at test time, we need to synthesize color strokes for the training data. We generate synthetic strokes based on the colors in the ground-truth image.

To emulate arbitrary user behaviors, we blur the ground-truth image and sample a random number of color strokes of random length and thickness at random locations. We pick the ground-truth pixel color at the stroke starting point as the stroke color and continue to grow the stroke until the maximum length is reached.

When growing a stroke, if the difference between the current pixel color and the stroke color exceeds a certain threshold, we restart the stroke with a new color sampled at the current pixel. By randomizing various stroke parameters, we are able to synthesize color strokes similar to what users would draw during test time.

5.2. Guided Sketch Colorization

The goal here is to add color control to our sketch-based image synthesis pipeline. Our previous objective function still holds: we want the output to have the same content as the input (pixel and feature loss), and appear realistic (adversarial loss). Pixel loss is essential here as it forces the network to be more precise with color by paying more attention to the color strokes. We modify the training data by placing color strokes on top of the input sketches. We

Figure 7. Guided Image Colorization: a) grayscale input, b) original color image, c) deep colorization result [54], d) First and third rows: color strokes overlaid on top of the input (zoom in to see the color strokes). Second and fourth rows: colorization results.

then train the network as before using a parameter setting that emphasizes content loss and de-emphasizes adversarial loss, so that the results better satisfy color constraints (Section 7.2). Figure 6 shows the results of reconstructing bedroom and car scenes based on an input sketch and color strokes. Note that the colors of the strokes deviate a lot from the colors in the ground-truth image, nevertheless, the network is able to propagate the input color to the relevant regions respecting object boundaries.

5.3. Guided Image Colorization

Recent work [17, 54] explores training deep neural network models for the image colorization tasks. However, the selection of colors in the output is entirely up to the network. In this section, we investigate using color strokes (Section 5.1) to guide the colorization process. We generate training data by extracting a one-channel grayscale image from the ground-truth photo and combining it with the three-channel image containing color strokes.

Figure 7 shows various colorization results on a car image. Given a gray-scale image, our system synthesizes realistic looking cars based on strokes drawn with different colors at random locations. Note that most strokes are placed on the body of the car and therefore do not influence the
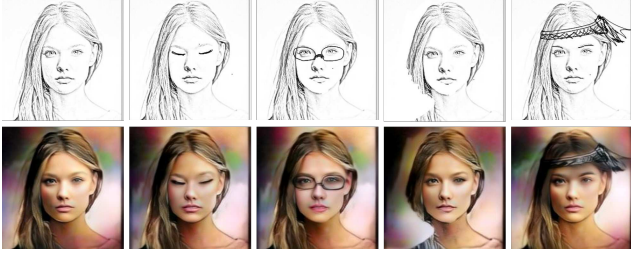
Figure 8. Interactive image editing. The user can incrementally modify the sketch to change the eyes, hair, and head decorations.

colorization of the other regions. Due to adversarial training, the sky is colored blue and the trees are colored green, regardless of the colors of the foreground object. Internally, the network learns to recognize semantic contents and therefore can put right colors in the relevant regions while at the same time satisfying user constraints wherever possible.

6. Interactive Image Generation Tools

Our network transforms an input sketch of resolution 256x256 to a photo-like result in 20ms. The real-time performance enables instant visual feedback after incremental edits in image generation and editing applications.

Using sketch and color strokes to enforce fine-grained control is useful for several design applications. For example, an interior designer can quickly sketch out rough shapes of the objects, specify colors in various regions and let our system fill in missing details and textures to generate a plausible bedroom scene. After seeing the result, the designer can interactively modify the shapes and colors of the objects and receive instant visual feedback. Figure 6 illustrates the potential design workflow. Similarly, a car designer can follow similar workflow to design new cars and test out the looks in different background settings.

Our portrait synthesis system could be a tool for artists to design virtual characters (see Figure 8). Based on the initial design, one can change the shape of eyes and/or hairstyle, add glasses and/or head decorations, etc. In addition to design, portrait reconstruction technology is useful for forensic purposes, e.g. law enforcement visualization of suspects [14].

7. Network Training Details

With the unpredictable nature of adversarial training, we find it helpful to separate the training into two stages.

7.1. Optimizing for Content Loss

In the first stage, we set the adversarial weight $\mathbf{w_{adv}}$ from equation 2 to 0 and let the network focus on minimizing the content loss which is a combination of pixel and feature loss. To enforce a fine-grained control using the input sketch, we choose the ReLU2-2 layer of the VGG-19

net [44] to compute the feature loss, since higher level feature representations tend to encourage the network to ignore important details such as the exact locations of the pupils.

We set the weights of pixel loss and feature loss $\mathbf{w_p}$, $\mathbf{w_f}$ to 1, and the weight of TV loss $\mathbf{w_{tv}}$ to 1e-5. We train the network for around 3 epochs using a batch size of 32 before moving on to the second stage of the training.

7.2. Adding Adversarial Loss

Given the network pretrained for content loss, we fine tune it with different loss settings for different applications. For photo reconstruction from gray-scale sketches (Section 4), we turn off the pixel loss, keep the feature loss and add the adversarial loss with the following weight setting, $\mathbf{w_f} = 1, \mathbf{w_p} = 0, \mathbf{w_{tv}} = 0, \mathbf{w_{adv}} \approx 1e8$. For colorization applications (Section 5), we emphasize the feature and pixel loss and de-emphasize the adversarial loss, so that the output better follows the color controls, $\mathbf{w_f} = 10, \mathbf{w_p} = 1, \mathbf{w_{tv}} = 0, \mathbf{w_{adv}} \approx 1e5$.

We train the adversarial discriminator alongside our generative network for three epochs using a learning rate between 1e-5 and 1e-6.

8. Conclusion and Future Work

In this paper, we propose a deep generative framework that enables two types of user controls to guide the result generation – using sketch to guide high-level visual structure and using sparse color strokes to control color pattern.

Despite the promising results, our current system suffers from several limitations. First, we sometimes observe blurry boundaries between object parts or regions of different colors which diminishes the overall realism of the results.

Figure 6 shows the color leaking problem on the car results, where the color of the car's hood leaks into the background. Second, our system struggles between strictly following color/sketch controls and minimizing adversarial loss. In other words, adversarial loss prohibits the generated images from taking uncommon colors and shapes. If the user specifies a rare color, for example, purple for car, red for trees, our network will map it to a different color deemed more realistic by the adversarial loss.

Going forward, we would like to investigate how to improve the results by encouraging sharp color boundaries and finding systematic ways to deal with rare control signals.

Acknowledgments

References

[1] Convert photo to line drawing. www.youtube.com/watch?v=Gyu2yPwiQvA, 2012. 5

[2] Create filter gallery photocopy effect with single step in photoshop. www.youtube.com/watch?v=QNmniB_5Nz0/, 2016. 5

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009. 2

[4] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural Photo Editing with Introspective Adversarial Networks. *ArXiv e-prints*, Sept. 2016. 2

[5] A. J. Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv preprint arXiv:1603.01768*, 2016. 2

[6] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics (TOG)*, 28(5):124, 2009. 2

[7] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016. 4

[8] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1, 2

[9] A. A. Efros and T. K. Leung. Texture synthesis by nonparametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038 vol.2, 1999. 2

[10] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Trans. Graph.*, 31(4), July 2012. 3

[11] 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*, pages 2414–2423, 2016. 3, 5

[12] 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*, pages 2672–2680, 2014. 1, 2, 4

[13] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015. 1, 2

[14] Y. Güçlütürk, U. Güçlü, R. van Lier, and M. A. van Gerven. Convolutional sketch inversion. In *Proceeding of the ECCV workshop on VISART Where Computer Vision Meets Art*, 2016. 1, 2, 3, 4, 6, 8

[15] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, 2007. 2

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[17] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4), 2016. 2, 3, 7

[18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016. 1, 2, 3

[19] J. Johnson, A. Alahi, and L. Fei-Fei. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*, pages 694–711. Springer International Publishing, Cham, 2016. 4

[20] C. Kaae Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised MAP Inference for Image Super-resolution. *ArXiv e-prints*, Oct. 2016. 2

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, 2014. 1, 2

[22] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3):3, August 2007. 2

[23] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[24] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 2

[25] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, 2009. 1

[26] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 689–694. ACM, 2004. 3

[27] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[28] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[29] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 5

[30] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 39–46, 1995. 2

[31] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems 29*. 2016. 2

[32] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. In *arXiv pre-print*. 2016. 2

[33] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. distill.pub/2016/deconv-checkerboard/, 2016. 4

[34] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33th International Conference on Machine Learning, ICML 2016*, 2016. 2

[35] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. 2016. 2, 4

[36] Y. Qu, T.-T. Wong, and P.-A. Heng. Manga colorization. *ACM Transactions on Graphics (SIGGRAPH 2006 issue)*, 25(3):1214–1220, July 2006. 3

[37] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1, 2

[38] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *NIPS*, 2016. 2, 3

[39] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016. 2

[40] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, pages 234–241. 2015. 3

[41] R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009. 1

[42] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016. 3

[43] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. *ACM Transactions on Graphics (SIGGRAPH)*, 35(4), 2016. 3

[44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 8

[45] D. Sỳkora, J. Dingliana, and S. Collins. Lazybrush: Flexible painting tool for hand-drawn cartoons. In *Computer Graphics Forum*, volume 28, pages 599–608. Wiley Online Library, 2009. 3

[46] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances In Neural Information Processing Systems*, pages 4790–4798, 2016. 1, 2

[47] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[48] X. Wang and X. Tang. Face photo-sketch synthesis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1955–1967, 2009. 6

[49] H. WinnemöLler, J. E. Kyprianidis, and S. C. Olsen. Xdog: an extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 36(6):740–753, 2012. 5

[50] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 1, 2

[51] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016. 2

[52] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixellevel domain transfer. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[53] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. 5

[54] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016. 1, 2, 3, 7

[55] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2

[56] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 3