TextureGAN: Controlling Deep Image Synthesis with Texture Patches

Wenqi Xian [†1] Patsorn Sangkloy [†1] Varun Agrawal [1] Amit Raj [1]
Jingwan Lu [2] Chen Fang [2] Fisher Yu [3] James Hays [1,4]
[1]Georgia Institute of Technology [2]Adobe Research [3]UC Berkeley [4]Argo AI

Figure 1. With TextureGAN, one can generate novel instances of common items from hand drawn sketches and simple texture patches. You can now be your own fashion guru! Top row: Sketch with texture patch overlaid. Bottom row: Results from TextureGAN.

Abstract

In this paper, we investigate deep image synthesis guided by sketch, color, and texture. *Previous image synthesis methods can be controlled by sketch and color strokes but we are the first to examine texture control. We allow a user to place a texture patch on a sketch at arbitrary locations and scales to control the desired output texture. Our generative network learns to synthesize objects consistent with these texture suggestions. To achieve this, we develop a local texture loss in addition to adversarial and content loss to train the generative network. We conduct experiments using sketches generated from real images and textures sampled from a separate texture database and results show that our proposed algorithm is able to generate plausible images that are faithful to user controls. Ablation studies show that our proposed pipeline can generate more realistic images than adapting existing methods directly.*

1. Introduction

One of the "Grand Challenges" of computer graphics is to allow *anyone* to author realistic visual content. The traditional 3d rendering pipeline can produce astonishing and realistic imagery, but only in the hands of talented and trained artists. The idea of short-circuiting the traditional 3d modeling and rendering pipeline dates back at least 20 years to image-based rendering techniques [33]. These techniques and later "image-based" graphics approaches focus on re-using image content from a database of training images [22]. For a limited range of image synthesis and editing scenarios, these non-parametric techniques allow non-experts to author photorealistic imagery.

In the last two years, the idea of direct image synthesis without using the traditional rendering pipeline has gotten significant interest because of promising results from deep network architectures such as Variational Autoencoders (VAEs) [21] and Generative Adversarial Networks (GANs) [11]. However, there has been little investigation of fine-grained *texture* control in deep image synthesis (as opposed to coarse texture control through "style transfer" methods [9]).

In this paper we introduce TextureGAN, the first deep image synthesis method which allows users to control object texture. Users "drag" one or more example textures onto sketched objects and the network realistically applies these textures to the indicated objects.

This "texture fill" operation is difficult for a deep network to learn for several reasons: (1) Existing deep networks aren't particularly good at synthesizing high-resolution texture details even without user constraints. Typical results from recent deep image synthesis methods are at low resolution (e.g. 64x64) where texture is not prominent or they are higher resolution but relatively flat (e.g. birds with sharp boundaries but few fine-scale de-

[†] indicates equal contribution

tails). (2) For TextureGAN, the network must learn to propagate textures to the relevant object boundaries – it is undesirable to leave an object partially textured or to have the texture spill into the background. To accomplish this, the network must implicitly segment the sketched objects *and* perform texture synthesis, tasks which are individually difficult. (3) The network should additionally learn to foreshorten textures as they wrap around 3d object shapes, to shade textures according to ambient occlusion and lighting direction, and to understand that some object parts (handbag clasps) are not to be textured but should occlude the texture. These texture manipulation steps go beyond traditional texture synthesis in which a texture is assumed to be stationary. To accomplish these steps the network needs a rich implicit model of the visual world that involves some partial 3d understanding.

Fortunately, the difficulty of this task is somewhat balanced by the availability of training data. Like recent unsupervised learning methods based on colorization [47, 23], training pairs can be generated from unannotated images. In our case, input training sketches and texture suggestions are automatically extracted from real photographs which in turn serve as the ground truth for initial training. We introduce *local* texture loss to further fine-tune our networks to handle diverse textures unseen on ground truth objects.

We make the following contributions:

- We are the first to demonstrate the plausibility of fine-grained texture control in deep image synthesis. In concert with sketched object boundaries, this allows non-experts to author realistic visual content. Our network is feed-forward and thus can run interactively as users modify sketch or texture suggestions.

- We propose a "drag and drop" texture interface where users place particular textures onto sparse, sketched object boundaries. The deep generative network directly operates on these localized texture patches and sketched object boundaries.

- We explore novel losses for training deep image synthesis. In particular we formulate a local texture loss which encourages the generative network to handle new textures never seen on existing objects.

2. Related Work

Image Synthesis. Synthesizing natural images has been one of the most intriguing and challenging tasks in graphics, vision, and machine learning research. Existing approaches can be grouped into non-parametric and parametric methods. On one hand, non-parametric approaches have a long-standing history. They are typically data-driven or example-based, i.e., directly exploit and borrow existing image pixels for the desired tasks [1, 3, 6, 13, 33]. Therefore, non-parametric approaches often excel at generating realistic results while having limited generalization ability, i.e., being restricted by the limitation of data and examples, e.g., data bias and incomplete coverage of long-tail distributions. On the other hand, parametric approaches, especially deep learning based approaches, have achieved promising results in recent years. Different from non-parametric methods, these approaches utilize image datasets as training data to fit deep parametric models, and have shown superior modeling power and generalization ability in image synthesis [11, 21], e.g., hallucinating diverse and relatively realistic images that are different from training data.

Generative Adversarial Networks (GANs) [11] are a type of parametric method that has been widely applied and studied for image synthesis. The main idea is to train paired generator and discriminator networks jointly. The goal of the discriminator is to classify between 'real' images and generated 'fake' images. The generator aims to fool the discriminator by generating images which are indistinguishable from real images. Once trained, the generator can be used to synthesize images when seeded with a noise vector. Compared to the blurry and low-resolution outcome from other deep learning methods [21, 4], GAN-based methods [35, 32, 17, 49] generate more realistic results with richer local details and of higher resolution.

Controllable Image Synthesis and Conditional GANs. Practical image synthesis tools require human-interpretable controls. These controls could range from high-level attributes, such as object classes [34], object poses [4], natural language descriptions [36], to fine-grained details, such as segmentation masks [17], sketches [37, 12], color scribbles [37, 48], and cross-domain images [9, 44].

While the 'vanilla' GAN is able to generate realistic looking images from noise, it is not easily controllable. *Conditional* GANs are models that synthesize images based on input modalities other than simple noise, thus offering more control over the generated results. Compared to vanilla GANs, conditional GANs introduce additional discriminators or losses to guide generators to output images with desired properties, e.g., an object category discriminator [34], a discriminator to judge visual-text association [36], or a simple pixel-wise loss between generated images and target images [17].

It is worth highlighting several recent works on sketch or color-constrained deep image synthesis. Scribbler [37] takes as input a sketch and short color strokes, and generates realistically looking output that follows the input sketch and has color consistent with the color strokes. A similar system is employed for automatically painting cartoon images [29]. A user-guided interactive image colorization system was proposed in [48], offering users the control of

color when coloring or recoloring an input image. Distinct from these works, our system simultaneously supports richer user guidance signals including structural sketches, color patches, *and* texture swatches. Moreover, we examine new loss functions.

Texture Synthesis and Style Transfer. Texture synthesis and style transfer are two closely related topics in image synthesis. Given an input texture image, texture synthesis aims at generating new images with visually similar textures. Style transfer has two inputs – *content* and *style* images – and aims to synthesize images with the layout and structure of the content image and the texture of the style image. Non-parametric texture synthesis and style transfer methods typically resample provided example images to form the output [6, 5, 40, 14]. TextureShop [7] is similar to our method in that it aims to texture an object with a user-provided texture, but the technical approach is quite different. TextureShop uses non-parametric texture synthesis and shape-from-shading to foreshorten the texture so that it appears to follow the surface of a photographed object.

A recent deep style transfer method by Gatys et al. [8, 9] demonstrates that the correlations (i.e., Gram matrix) between features extracted from a pre-trained deep neural network capture the characteristics of textures well and showed promising results in synthesizing textures and transferring styles. Texture synthesis and style transfer are formalized as an optimization problem, where an output image is generated by minimizing a loss function of two terms, one of which measures content similarity between the input content image and the output, and the other measures style similarity between the input style and the output using the Gram matrix. Since the introduction of this approach by Gatys et al. [8, 9], there have been many works on improving the generalization [46, 15, 26], efficiency [39, 19] and controllability [10] of deep style transfer.

Several texture synthesis methods use GANs to improve the quality of the generated results. Li and Wand [25] use adversarial training to discriminate between real and fake textures based on a feature patch from the VGG network. Instead of operating on feature space, Jetchev et al. [18] and Bergman et al. [2] apply adversarial training at the pixel level to encourage the generated results to be indistinguishable from real texture. Our proposed texture discriminator in Section 3.2.1 differs from prior work by comparing a *pair* of patches from generated and ground truth textures instead of using a single texture patch. Intuitively, our discriminator is tasked with the fine-grained question of "is this the same texture?" rather than the more general "is this a valid texture?". Fooling such a discriminator is more difficult and requires our generator to synthesize not just realistic texture but also texture that is faithful to various input texture styles.

Similar to texture synthesis, image completion or inpainting methods also show promising results using GANs.

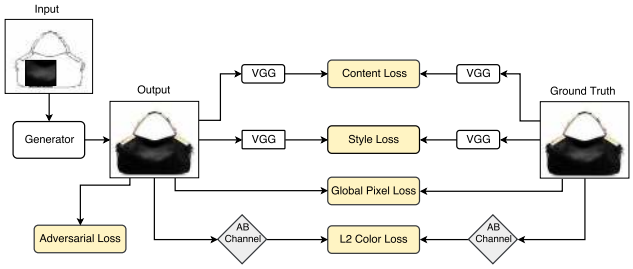

Figure 2. TextureGAN pipeline for the ground-truth pre-training (section 3.1)

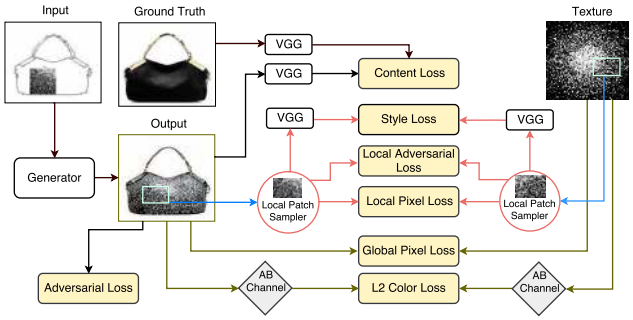

Figure 3. TextureGAN pipeline for the external texture fine-tuning (section 3.2)

Our task has similarities to the image completion problem, which attempts to fill in missing regions of an image, although our missing area is significantly larger and partially constrained by sketch, color, or texture. Similar to our approach, Yang et al. [43] computes texture loss between patches to encourage the inpainted region to be faithful to the original image regions. However, their texture loss only accounts for similarity in feature space. Our approach is similar in spirit to Iizuka et al. [16], which proposes using both global and local discriminators to ensure that results are both realistic and consistent with the image context, whereas our local discriminator is instead checking texture similarity between input texture patch and output image.

3. TextureGAN

We seek an image synthesis pipeline that can generate natural images based on an input *sketch* and some number of user-provided *texture* patches. Users provide rough sketches that outline the desired objects to control the generation of semantic content, e.g. object type and shape, but sketches do not contain enough information to guide the generation of texture details, materials, or patterns. To guide the generation of fine-scale details, we want users to somehow control texture properties of objects and scene elements.

Towards this goal, we introduce **TextureGAN**, a conditional generative network that learns to generate realistic images from input sketches with overlaid textures. We ar-

gue that instead of providing an unanchored texture sample, users can more precisely control the generated appearance by directly placing small texture patches over the sketch, since locations and sizes of the patches provide hints about the object appearance desired by the user. In this setup, the user can 'drag' rectangular texture patches of arbitrary sizes onto different sketch regions as additional input to the network. For example, the user can specify a striped texture patch for a shirt and a dotted texture patch for a skirt. The input patches guide the network to propagate the texture information to the relevant regions respecting semantic boundaries (e.g. dots should appear on the skirt but not on the legs).

A major challenge for a network learning this task is the uncertain pixel correspondence between the input texture and the unconstrained sketch regions. To encourage the network to produce realistic textures, we propose a local texture loss (Section 3.2) based on a texture discriminator and a Gram matrix style loss. This not only helps the generated texture follow the input faithfully, but also helps the network learn to propagate the texture patch and synthesize new texture.

TextureGAN also allows users to more precisely control the colors in the generated result. One limitation of previous color control with GANs [37] is that the input color constraints in the form of $\mathbf{RGB}$ need to fight with the network's understanding about the semantics, e.g., bags are mostly black and shoes are seldom green. To address this problem, we train the network to generate images in the $\mathbf{Lab}$ color space. We convert the groundtruth images to $\mathbf{Lab}$, enforce the content, texture and adversarial losses only on the $\mathbf{L}$ channel, and enforce a separate color loss on the $\mathbf{ab}$ channels. We show that combining the controls in this way allows the network to generate realistic photos closely following the user's color and texture intent without introducing obvious visual artifacts.

We use the network architecture proposed in Scribbler [37] with additional skip connections. Details of our network architecture are included in the supplementary material. We use a 5-channel image as input to the network. The channels support three different types of controls – one channel for sketch, two channels for texture (one intensity and one binary location mask), and two channels for color. Section 4.2 describes the method we used to generate each input channel of the network.

We first train TextureGAN to reproduce ground-truth shoe, handbag, and clothes photos given synthetically sampled input control channels. We then generalize TextureGAN to support a broader range of textures and to propagate unseen textures better by fine-tuning the network with a separate texture-only database.

3.1. Ground-truth Pre-training

We aim to propagate the texture information contained in small patches to fill in an entire object. As in Scribbler [37], we use feature and adversarial losses to encourage the generation of realistic object structures. However, we find that these losses alone cannot reproduce fine-grained texture details. Also, Scribbler uses pixel loss to enforce color constraints, but fails when the input color is rare for that particular object category. Therefore, we redefine the feature and adversarial losses and introduce new losses to improve the replication of texture details and encourage precise propagation of colors. For initial training, we derive the network's input channels from ground-truth photos of objects. When computing the losses, we compare the generated images with the ground-truth. Our objective function consists of multiple terms, each of which encourages the network to focus on different image aspects. Figure 2 shows our pipeline for the ground-truth pre-training.

Feature Loss $\mathcal{L}_F$. It has been shown previously that the features extracted from middle layers of a pre-trained neural network, VGG-19 [38], represent high-level semantic information of an image [12, 37]. Given a rough outline sketch, we would like the generated image to loosely follow the object structures specified by the sketch. Therefore, we decide to use a deeper layer of VGG-19 for feature loss (relu 4_2). To focus the feature loss on generating structures, we convert both the ground-truth image and the generated image from $\mathbf{RGB}$ color space to $\mathbf{Lab}$ and generate grayscale images by repeating the $\mathbf{L}$ channel values. We then feed the grayscale image to VGG-19 to extract features. The feature loss is defined as the L2 difference in the feature space. During back propagation, the gradients passing through the L channel of the output image are averaged from the three channels of the VGG-19 output.

Adversarial Loss $\mathcal{L}_{ADV}$. In recent work, the concept of adversarial training has been adopted in the context of image to image translation. In particular, one can attach a trainable discriminator network at the end of the image translation network and use it to constrain the generated result to lie on the training image manifold. Previous work proposed to minimize the adversarial loss (loss from the discriminator network) together with other standard losses (pixel, feature losses, etc). The exact choice of losses depends on the different applications [37, 17, 12]. Along these lines, we use adversarial loss on top of feature, texture and color losses. The adversarial loss pushes the network towards synthesizing sharp and realistic images, but at the same time constrains the generated images to choose among typical colors in the training images. The network's understanding about color sometimes conflicts with user's color constraints, e.g. a user provides a rainbow color constraint for a handbag, but the adversarial network thinks it looks fake and discourages the generator from producing

Figure 4. The effect of texture loss and adversarial loss. a) The network trained using all proposed losses can effectively propagate textures to most of the foreground region; b) Removing adversarial loss leads to blurry results; c) Removing texture loss harms the propagation of textures.

such output. Therefore, we propose applying the adversarial loss $\mathcal{L}_{adv}$ only on grayscale image (the $\mathbf{L}$ channel in $\mathbf{Lab}$ space). The discriminator is trained to disregard the color but focus on generating sharp and realistic details. The gradients of the loss only flow through the $\mathbf{L}$ channel of the generator output. This effectively reduces the search space and makes GAN training easier and more stable. We perform the adversarial training using the techniques proposed in DCGAN [35] with the modification proposed in LSGAN [32]. LSGAN proposed replacing the cross entropy loss in the original GAN with least square loss for higher quality results and stable training.

Style Loss $\mathcal{L}_S$. In addition to generating the right content following the input sketch, we would also like to propagate the texture details given in the input texture patch. The previous feature and adversarial losses sometimes struggle to capture fine-scale details, since they focus on getting the overall structure correct. Similar to deep learning based texture synthesis and style transfer work [8, 9], we use style loss to specifically encourage the reproduction of texture details, but we apply style loss on the $\mathbf{L}$ channel only. We adopt the idea of matching the Gram matrices (feature correlations) of the features extracted from certain layers of the pretrained classification network (VGG-19). The Gram matrix $\mathcal{G}^l_{ij} \in \mathcal{R}^{N_l \times N_l}$ is defined as:

$$\mathcal{G}^l_{ij} = \sum_k \mathcal{F}^l_{ik} \mathcal{F}^l_{jk} \tag{1}$$

where, N_l is the number of feature maps at network layer l, $\mathcal{F}^l_{ik}$ is the activation of the ith filter at position k in layer l. We use two layers of the VGG-19 network (relu3_2, relu4_2) to define our style loss.

Pixel Loss $\mathcal{L}_P$. We find that adding relatively weak L2 pixel loss on the $\mathbf{L}$ channel stabilizes the training and leads to the generation of texture details that are more faithful to the user's input texture patch.

Color Loss $\mathcal{L}_C$. All losses above are applied only on the $\mathbf{L}$ channel of the output to focus on generating sketch-conforming structures, realistic shading, and sharp high-frequency texture details. To enforce the user's color constraints, we add a separate color loss that penalizes the L2 difference between the $\mathbf{ab}$ channels of the generated result and that of the ground-truth.

Our combined objective function is defined as:

$$\mathcal{L} = \mathcal{L}_F + \mathbf{w}_{ADV}\mathcal{L}_{ADV} + \mathbf{w}_S\mathcal{L}_S + \mathbf{w}_P\mathcal{L}_P + \mathbf{w}_C\mathcal{L}_C \tag{2}$$

3.2. External Texture Fine-tuning

One problem of training with "ground-truth" images is that it is hard for the network to focus on reproducing low-level texture details due to the difficulty of disentangling the texture from the content within the same image. For example, we do not necessarily have training examples of the same object with different textures applied which might help the network learn the factorization between structure and texture. Also, the Gram matrix-based style loss can be dominated by the feature loss since both are optimized for the same image. There is not much room for the network to be creative in hallucinating low-level texture details, since it tends to focus on generating high-level structure, color, and patterns. Finally, many of the ground-truth texture patches contain smooth color gradients without rich details. Trained solely on those, the network is likely to ignore "hints" from an unseen input texture patch at test time, especially if the texture hint conflicts with information from the sketch. As a result, the network often struggles to propagate high-frequency texture details in the results especially for textures that are rarely seen during training.

To train the network to propagate a broader range of textures, we fine-tune our network to reproduce and propagate textures *for which we have no ground truth output*. To do this, we introduce a new local texture loss and adapt our existing losses to encourage faithfulness to a *texture* rather than faithfulness to a ground truth output *object photo*. We use all the losses introduced in the previous sections except the global style loss $\mathcal{L}_S$. We keep the *feature and adversarial losses*, $\mathcal{L}_F, \mathcal{L}_{ADV}$, unchanged, but modify the *pixel and color losses*, $\mathcal{L}'_P, \mathcal{L}'_C$, to compare the generated result with the entire input texture from which input texture patches are extracted. Figure 3 shows our pipeline for the external texture fine-tuning. To prevent color and texture bleeding, the losses are applied only on the foreground object, as approximated by a segmentation mask (Section 4.1).

Figure 5. Effect of proposed *local* texture losses. Results from the ground-truth model a) without any local losses, b) with local pixel loss, c) with local style loss, d) with local adversarial loss. With local adversarial loss, the network tends to produce more consistent texture throughout the object.

3.2.1 Local Texture Loss

To encourage better propagation of texture, we propose a **local texture loss** $\mathcal{L}_t$, that is only applied to small local regions of the output image. We randomly sample n patches of size $s \times s$ from the generated result and the input texture I_t from a separate texture database. We only sample patches which fall inside an estimated foreground segmentation mask R (section 4.1). The local texture loss $\mathcal{L}_t$ is composed of three terms:

$$\mathcal{L}_t = \mathcal{L}_s + \mathbf{w}_p \mathcal{L}_p + \mathbf{w}_{adv} \mathcal{L}_{adv} \tag{3}$$

Local Adversarial Loss $\mathcal{L}_{adv}$. We introduce a local adversarial loss that decides whether a pair of texture patches have the same textures. We train a local texture discriminator $\mathbf{D_{txt}}$ to recognize a pair of cropped patches from the same texture as a positive example ($D_{txt}(\cdot) = 1$), and a pair of patches from different textures as a negative example ($D_{txt}(\cdot) = 0$).

Let $h(x, R)$ be a cropped patch of size $s \times s$ from image x based on segmentation mask R. Given a pair of cropped patches $(PG_i, PT_i) = (h(\mathbf{G}(x_i), R_i), h(I_t, R_i))$, we define L_{adv} as follows:

$$\mathcal{L}_{adv} = -\sum_i (\mathbf{D_{txt}}(PG_i, PT_i) - 1)^2 \tag{4}$$

Local Style Loss $\mathcal{L}_s$ and Pixel Loss $\mathcal{L}_p$. To strengthen the texture propagation, we also use Gram matrix-based style loss and L2 pixel loss on the cropped patches.

While performing the texture fine-tuning, the network is trying to adapt itself to understand and propagate new types of textures, and might 'forget' what it learnt from the ground-truth pretraining stage. Therefore, when training on external textures, we mix in iterations of ground-truth training fifty percent of the time.

Our final objective function becomes:

$$\mathcal{L} = \mathcal{L}_F + \mathbf{w}_{ADV} \mathcal{L}_{ADV} + \mathbf{w}_P \mathcal{L}'_P + \mathbf{w}_C \mathcal{L}'_C + \mathcal{L}_t \tag{5}$$

4. Training Setup

We train TextureGAN on three object-centric datasets – **handbags** [49], **shoes** [45] and **clothes** [27, 28, 30, 31]. Each photo collection contains large variations of colors, materials, and patterns. These domains are also chosen so that we can demonstrate plausible product design applications. For supervised training, we need to generate (input, output) image pairs. For the output of the network, we convert the ground-truth photos to **Lab** color space. For the input to the network, we process the ground-truth photos to extract 5-channel images. The five channels include one channel for the binary sketch, two channels for the texture (intensities and binary location masks), and two channels for the color controls.

In this section, we describe how we obtain segmentation masks used during training, how we generate each of the input channels for the ground-truth pre-training, and how we utilize the separate texture database for the network fine-tuning. We also provide detailed training procedures and parameters.

4.1. Segmentation Mask

For our local texture loss, we hope to encourage samples of output texture to match samples of input texture. But the output texture is localized to particular image regions (e.g. the interior of objects) so we wouldn't want to compare a background patch to an input texture. Therefore we only sample patches from within the foreground. Our handbag and shoe datasets are product images with consistent, white backgrounds so we simply set the white pixels as background pixels. For clothes, the segmentation mask is already given in the dataset [24, 28]. With the clothes segmentation mask, we process the ground-truth photos to white out the background. Note that segmentation masks are *not used* at test time.

4.2. Data Generation for Pre-training

Sketch Generation. For handbags and shoes, we generate sketches using the deep edge detection method used in pix2pix [42, 17]. For clothes, we leverage the clothes parsing information provided in the dataset [27, 28]. We apply Canny edge detection on the clothing segmentation mask to extract the segment boundaries and treat them as a sketch. We also apply xDoG [41] on the clothes image to obtain more variation in the training sketches. Finally, we mix in additional synthetic sketches generated using the methods proposed in Scribbler [37].

Texture Patches. To generate input texture constraints, we randomly crop small regions within the foreground objects of the ground-truth images. We randomly choose the patch location from within the segmentation and randomize the patch size. We convert each texture patch to the **Lab** color space and normalize the pixels to fall into 0-1 range. For each image, we randomly generate one or two texture patches. For clothes, we extract texture patches from one of the following regions – top, skirt, pant, dress, or bag. We compute a binary mask to encode the texture patch location.

4.3. Data Generation for Fine-tuning

To encourage diverse and faithful texture reproduction, we fine-tune TextureGAN by applying external texture patches from a leather-like texture dataset. We queried "leather" in Google and manually filtered the results to 130 high resolution leather textures. From this clean dataset, we sampled roughly 50 crops of size 256x256 from each image to generate a dataset of 6,300 leather-like textures. We train our models on leather-like textures since they are commonly seen materials for handbags, shoes and clothes and contain large appearance variations that are challenging for the network to propagate.

4.4. Training Details

For **pre-training**, we use the following parameters on all datasets. $\mathbf{w}_{ADV} = 1$, $\mathbf{w}_S = 0.1$, $\mathbf{w}_P = 10$ and $\mathbf{w}_C = 100$. We use the Adam optimizer [20] with learning rate 1e-2.

For **fine-tuning**, we optimize all the losses at the same time but use different weight settings. $\mathbf{w}_{ADV} = 1e4$, $\mathbf{w}_S = 0$, $\mathbf{w}_P = 1e2$, $\mathbf{w}_C = 1e3$, $\mathbf{w}_s = 10$, $\mathbf{w}_p = 0.01$, and $\mathbf{w}_{adv} = 7e3$. We also decrease the learning rate to 1e-3. We train most of the models at input resolution of 128x128 except one clothes model at the resolution of 256x256 (Figure 8).

5. Results and Discussions

Ablation Study. Keeping other settings the same, we train networks using different combinations of losses to analyze how they influence the result quality. In Figure 4, given the input sketch, texture patch and color patch (first column), the network trained with the complete objective function (second column) correctly propagates the color and texture to the entire handbag. If we turn off the texture loss (fourth column), the texture details within the area of the input patch are preserved, but difficult textures cannot be fully propagated to the rest of the bag. If we turn off the adversarial loss (third column), texture is synthesized, but that texture is not consistent with the input texture. Our ablation experiment confirms that style loss alone is not sufficient to encourage texture propagation motivating our local patch-based texture loss (Section 3.2.1).

Figure 6. Results on held out shoes and handbags sketches [152x152]. On the far left is the "ground truth" photo from which the sketch was synthesized. On the first result column, a texture patch is also sampled from the original shoe. We show three additional results with diverse textures.

External Texture Fine-tuning Results. We train TextureGAN on three datasets – shoes, handbags, and clothes – with increasing levels of structure complexity. We notice that for object categories like shoes that contain limited structure variations, the network is able to quickly generate realistic shading and structures and focus its remaining capacity for propagating textures. The texture propagation on the shoes dataset works well even without external texture fine-tuning. For more sophisticated datasets like handbags and clothes, external texture fine-tuning is critical for the propagation of difficult textures that contain sharp regular structures, such as stripes.

Figure 5 demonstrates how external texture fine-tuning with our proposed texture loss can improve the texture consistency and propagation.

The "ground truth" pre-trained model is faithful to the input texture patch in the output only directly under the patch and does not propagate it throughout the foreground region. By fine-tuning the network with texture examples and enforcing local style loss, local pixel loss, and local texture loss we nudge the network to apply texture consistently

Figure 7. Results for shoes and handbags on different textures. Odd rows: input sketch and texture patch. Even rows: generated results.

Figure 8. Applying multiple texture patches on the sketch. Our system can also handle multiple texture inputs and our network can follow sketch contours and expand the texture to cover the sketched object.

Figure 9. Results on human-drawn sketches. Sketch images from olesiaagudova - stock.adobe.com

across the object. With local style loss (column c) and local texture discriminator loss (column d), the networks are able to propagate texture better than without fine-tuning (column a) or just local pixel loss (column b). Using local texture discriminator loss tends to produce more visually similar result to the input texture than style loss.

Figures 6 and 7 show the results of applying various texture patches to sketches of handbags and shoes. These results are typical of test-time result quality. The texture elements in the camera-facing center of the bags tend to be larger than those around the boundary. Textures at the bottom of the objects are often shaded darker than the rest, consistent with top lighting or ambient occlusion. Note that even when the input patch goes out of the sketch boundary, the generated texture follow the boundary exactly.

Figure 8 shows results on the clothes dataset trained at a resolution of 256x256. The clothes dataset contains large variations of structures and textures, and each image in the dataset contains multiple semantic regions. Our network can handle multiple texture patches placed on different parts of the clothes (bottom left). The network can propagate the textures within semantic regions of the sketch while respecting the sketch boundaries.

Figure 9 shows results on human-drawn handbags. These drawings differ from our synthetically generated training sketches but the results are still high quality.

6. Conclusion

We have presented an approach for controlling deep image synthesis with input sketch and texture patches. With this system, a user can sketch the object structure and precisely control the generated details with texture patches. TextureGAN is feed-forward which allows users to see the effect of their edits in real time. By training TextureGAN with local texture constraints, we demonstrate its effectiveness on sketch and texture-based image synthesis. TextureGAN also operates in **Lab** color space, which enables separate controls on color and content. Furthermore, our results on fashion datasets show that our pipeline is able to handle a wide variety of texture inputs and generates texture compositions that follow the sketched contours. In the future, we hope to apply our network on more complex scenes.

Acknowledgments

This work is supported by a Royal Thai Government Scholarship to Patsorn Sangkloy and NSF award 1561968.

References

[1] 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

[2] U. Bergmann, N. Jetchev, and R. Vollgraf. Learning texture manifolds with the periodic spatial gan. *arXiv preprint arXiv:1705.06566*, 2017. 3

[3] T. Chen, M. ming Cheng, P. Tan, A. Shamir, and S. min Hu. Sketch2photo: internet image montage. *ACM SIGGRAPH Asia*, 2009. 2

[4] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. *CoRR*, abs/1411.5928, 2014. 2

[5] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001. 3

[6] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999. 2, 3

[7] H. Fang and J. C. Hart. Textureshop: Texture synthesis as a photograph editing tool. *ACM Trans. Graph.*, 23(3):354–359, Aug. 2004. 3

[8] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015. 3, 5

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, June 2016. 1, 2, 3, 5

[10] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. *arXiv preprint arXiv:1611.07865*, 2016. 3

[11] 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

[12] 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. 2, 4

[13] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 4. ACM, 2007. 2

[14] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001. 3

[15] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *arXiv preprint arXiv:1703.06868*, 2017. 3

[16] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)*, 36(4):107:1–107:14, 2017. 3

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016. 2, 4, 6

[18] N. Jetchev, U. Bergmann, and R. Vollgraf. Texture synthesis with spatial generative adversarial networks. *arXiv preprint arXiv:1611.08207*, 2016. 3

[19] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 3

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 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. 1

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

[24] C. Lassner, G. Pons-Moll, and P. V. Gehler. A generative model for people in clothing. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 6

[25] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016. 3

[26] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. *arXiv preprint arXiv:1703.01664*, 2017. 3

[27] X. Liang, S. Liu, X. Shen, J. Yang, L. Liu, J. Dong, L. Lin, and S. Yan. Deep human parsing with active template regression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(12):2402–2414, Dec 2015. 6

[28] X. Liang, C. Xu, X. Shen, J. Yang, S. Liu, J. Tang, L. Lin, and S. Yan. Human parsing with contextualized convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1386–1394, 2015. 6

[29] Y. Liu, Z. Qin, Z. Luo, and H. Wang. Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks. *arXiv preprint arXiv:1705.01908*, 2017. 2

[30] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[31] Z. Liu, S. Yan, P. Luo, X. Wang, and X. Tang. Fashion landmark detection in the wild. In *European Conference on Computer Vision (ECCV)*, 2016. 6

[32] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016. 2, 5

[33] 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*, pages 39–46. ACM, 1995. 1, 2

[34] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016. 2

[35] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2, 5

[36] 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*, volume 3, 2016. 2

[37] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. *Computer Vision and Pattern Recognition, CVPR*, 2017. 2, 4, 6

[38] 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. 4

[39] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*, 2016. 3

[40] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000. 3

[41] 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. 6

[42] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015. 6

[43] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3

[44] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016. 2

[45] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014. 6

[46] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017. 3

[47] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 2

[48] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017. 2

[49] 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. 2, 6