Unpaired Skeleton-to-Photo Translation for Sketch-to-Photo Synthesis

Yuanzhe Gu
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/masters_theses_2

Recommended Citation
https://doi.org/10.7275/30957864 https://scholarworks.umass.edu/masters_theses_2/1242

This Open Access Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Masters Theses by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
UNPAIRED SKELETON-TO-PHOTO TRANSLATION FOR SKETCH-TO-PHOTO SYNTHESIS

A Thesis Presented

by

YUANZHE GU

Submitted to the Graduate School of the University of Massachusetts Amherst in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2022

Electrical and Computer Engineering
UNPAIRED SKELETON-TO-PHOTO TRANSLATION FOR SKETCH-TO-PHOTO SYNTHESIS

A Thesis Presented
by
YUANZHE GU

Approved as to style and content by:

________________________________________
Marco F. Duarte, Chair

________________________________________
Mario Parente, Member

________________________________________
Hossein Pishro-Nik, Member

________________________________________
Christopher V. Hollot, Department Head
Electrical and Computer Engineering
ABSTRACT

UNPAIRED SKELETON-TO-PHOTO TRANSLATION FOR SKETCH-TO-PHOTO SYNTHESIS

SEPTEMBER 2022

YUANZHE GU

B.Sc., UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Marco F. Duarte

Sketch-to-photo synthesis usually faced the problem of lack of labeled data, so we propose some methods based on CycleGAN to train a model to translate sketch to photo with unpaired data. Our main contribution is a proposed Sketch-to-Skeleton-to-Image (SSI) method, which performs skeletonization on sketches to reduce variance on the sketch data. We also tried different representations of the skeleton and different models for our task. Experiment results show that the generated image quality has a negative correlation with the sparsity of the input data.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2. BACKGROUND</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.1 Generative adversarial networks</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.2 CycleGAN</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2.3 Sketch-Based Image Synthesis</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>2.4 Distance transformation</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>2.5 Skeleton Context Flux</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>2.6 Side-output multiscale model</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>3. CONTRIBUTIONS</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>3.1 Transforming a sketch to a skeleton</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>3.2 Basic structure: CycleGAN</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>3.3 Different presentations of skeletons: distance transformation and flux</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>3.4 Side-output model</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3.5 Side-input model</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>4. SIMULATION RESULTS</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>4.1 Computation Resources</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>4.2 Datasets</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>4.3 Evaluation Metrics</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>4.3.1 Frechet Inception Distance</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>4.3.2 Learned perceptual image patch similarity</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>4.4 Experimental Results</td>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>
5. CONCLUSIONS ................................................................. 28

BIBLIOGRAPHY ................................................................. 30
CHAPTER 1
INTRODUCTION

Sketch-to-photo translation is an approach to quickly visualize users’ intent on images. Due to the performance of generative adversarial networks [11] (GAN), many works adopt GAN to synthesize images from sketches [17, 22, 7]. However, the sketch-to-photo synthesis problem currently faces two serious issues: 1). It is hard to find paired freehand sketch and photo data. There are some works [17, 22, 27] that adopt unpaired image to image translation to deal with this problem, and we base our own work on these approaches. 2). There is significant distortion from freehand sketches to photos, especially for objects which have complex structures: the shape of the sketches and the real objects are always misaligned.

Although the sketches’ shape and other details are deformed from the objects, their structures still represent the users’ intention, which can be abstracted to skeletons. A skeleton is the medial axis of a foreground region; its width is 1 pixel, and the goal of the skeleton is to represent topology of a region. Figure 1.1 shows the distortion of shapes between sketches and objects and the similarity between their skeletons. Our approach is to apply Sketch-to-Skeleton-to-Image (SSI) translation, since freehand sketches usually share similar skeletons with real objects. Our method can correctly understand the object structure as shown in Figure 1.2.

Compared with other unsupervised sketch-to-photo translation methods, SSI has advantages in certain areas. For sketch-to-photo synthesis, we can define two kinds of objects; one is objects that have no single correct shape, like chairs or shoes, as shown in Figure 1.3, as they are usually designed by humans. Another kind of objects
Figure 1.1. Examples of freehand sketches and skeletons. (a) a free hand sketch of a horse. (b) skeleton of (a). (c) a real image of a horse. (d) skeleton of (c). The freehand sketch and real horse photo have large spatial distortion; however, their skeletons have good geometry correspondence.

Figure 1.2. Our approach for sketch-to-photo synthesis. In a first step we skeletonize the sketch, and then we apply unsupervised learning on skeleton to photo translation.

have a standard correct shape: they are usually animals or plants which are designed by nature. Most methods on sketch-to-photo translation are usually tested on the former class; on the contrary, our approach can perform better on images that have a determined shape. Another advantage of SSI is that skeletons remove noise that is drawn by mistake in sketches; the generated image quality would be more stable. A third advantage of a SSI approach is that skeletons can be obtained from many kind of sketches: usually sketches are collected for a variety of purposes, or in different conditions, like in the QuickDraw dataset [14], where sketches are drawn within 20 seconds, so the quality of these sketches are worse than other datasets, as can be seen in the example in Figure 1.1 (a).

This thesis is organized as follows: Chapter 2 provides necessary background on models and data processing methods that will be used in thesis. Chapter 3 introduces
Figure 1.3. Objects that do not have a particular correct shape. For this kind of objects, the corresponding sketches are widely varying. Thus, when the sketches are bad drawing, which means the shape of drawn sketch have large variant to what drawer want to draw, the generated photo still can be realistic for viewers. This provided an explanation as to why sketch-to-photo synthesis for chairs is easier than for horses.

our approach to sketch-to-photo transformation. Chapter 4 present simulation results showing our approach and comparing it to other methods. Chapter 5 provides the conclusion of the thesis.
Figure 1.4. (b) is the skeleton of (a), we can see that (a) is a bad drawing case of a zebra, but its skeleton is not affected by the distortion of (a).
CHAPTER 2
BACKGROUND

2.1 Generative adversarial networks

Generative adversarial networks (GAN) [11] are the most commonly used model for generation tasks. GAN consists of the discriminator and the generator: the generator is responsible for generating samples, and the discriminator is responsible for distinguishing between the generated samples and real samples. When training the GAN, the generator will try to generate samples that can fool the discriminator, so that while the discriminator’s performance is getting better, generators need to create more realistic samples. The learning objective function of GAN can be expressed as

$$
\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \quad (2.1)
$$

In [11], it has also been proved that optimizing (2.1) is equivalent to minimizing the Jensen-Shannon divergence between the generated samples’ distribution and the real samples’ distribution.

2.2 CycleGAN

CycleGAN is a kind of Generative Adversarial Networks that applies cycle-consistent loss. It consists of two generators, $G$ and $F$, and two discriminators, $D_X$ and $D_Y$. $G$ translates samples from domain $X$ to domain $Y$, and $F$ translates samples from domain $Y$ to domain $X$; $D_X$ distinguishes real samples in domain $X$ from samples
Figure 2.1. (a) Block diagram for the four models in CycleGAN. (b) and (c) show how cycle-consistent loss works. (Figure taken from [27])

translated by $F$; and vice versa for $D_Y$ and $G$ as shown in Figure 2.1(a). Then we have 2 adversarial losses:

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)} [\log (1 - D_Y(G(x)))]$$

(2.2)

$$L_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)} [\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)} [\log (1 - D_X(F(y)))]$$

(2.3)

In addition, the Cycle-consistent loss is based on a simple idea: for two well trained translators $G$ and $F$, $x \in X$, we have $x \xrightarrow{G} \hat{Y} \xrightarrow{F} \hat{x}$, $x$ should be close to $\hat{x}$. Thus we define the cycle-consistent loss as

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} [\| F(G(x)) - x \|_1] + \mathbb{E}_{y \sim p_{data}(y)} [\| G(F(y)) - y \|_1].$$

(2.4)

The full objective of CycleGAN is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F).$$

(2.5)
2.3 Sketch-Based Image Synthesis

In the early development of sketch-based image synthesis, some works [5, 8] use sketches as queries to retrieve photos and compose them together with background photograph. Similarly, PoseShop [6] uses human’s 2D skeletons as a query and retrieves composite images. After generative adversarial networks (GANs) [11] were proposed, the number of GAN-based models for sketch-to-photo translation has been increasing. While most of these GAN-based methods require paired data for training [7, 10, 12], some works [7, 16, 4, 12] use edge maps or outlines [10] that are extracted from images to obtain additional training data. CycleGAN [27] has been designed for unpaired image-to-image translation and applied on sketch-based image synthesis. The approach in [17] is based on CycleGAN and consists of a two-stage training for sketch-to-image translation. They generate grayscale images based on sketches first, and at the second stage they apply content enrichment to fill the grayscale images with colorful details. AODA [22] is another unsupervised learning model for sketch-to-image synthesis, which jointly learns photo-to-sketch and sketch-to-photo translations to solve the open-domain multi-class problem. AODA adds class labels in the sketch-to-photo Generator and applies an additional Discriminator to classify which class the generated image belong to, for open-domain class they use synthesized sketches to replace the missing data.

In comparison with these existing approaches, our proposed approach will apply skeletonization to sketches first, then learn a skeleton-to-image translation based on CycleGAN. To the best of our knowledge, we are the first to apply a skeleton representation into a GAN-based model for sketch-to-image translation. Compared with other methods, in SSI, the quality of the synthesized images has less dependency on the users’ drawing skill, which is especially important for natural objects that have a single correct shape. Even though the users’ drawing is deformed a lot from real objects, our skeleton-based image translation can still visualize users’ intention.
2.4 Distance transformation

We will use the efficient distance transformation algorithm proposed in [3]. For sparse binary images, it computes for each pixel the distance to the nearest feature pixel, as shown in Figure 2.2.

2.5 Skeleton Context Flux

DeepFlux [21], inspired by flux-based skeletonization methods [19], proposes a novel representation of the skeleton as shown in Figure 2.3. Formally, the flux \( F(p) \) is defined as:

\[
F(p) = \begin{cases} 
\frac{\overrightarrow{pN_p}}{|\overrightarrow{pN_p}|}, & p \in R_c, \\
(0,0), & p \in R_s \cup R_b, 
\end{cases}
\]  

(2.6)

where \( \overrightarrow{pN_p} \) is vector from a pixel to its nearest pixel in the skeleton, \( |\overrightarrow{pN_p}| \) is the vector's length, \( R_s \) is the region of skeleton, \( R_b \) is region of background, and \( R_c \) is the context region of context where the flux will be assigned.
Figure 2.3. Flux $F(p) \in \mathbb{R}^2$ is defined on the region $R_c$ which is equal to the dilated skeleton with the skeleton pixels removed. Each pixel $p \in R_c$ is assigned a vector towards its nearest pixel on skeleton $N_p$. For pixels on the skeleton, their flux is set to $(0,0)$. (Figure taken from [21])

The benefit of the Context Flux is similar to the distance transformation map in Section 2.4: compared with the skeleton, it adds more feature information, and makes the data less sparse; meanwhile a scope is easier to be generated by neural network compared with pixel width objects.

2.6 Side-output multiscale model

Side-output fusion was first proposed in HED (Holistically-Nested Edge Detection) for edge detection [23]; this method has recently been used in some skeleton extraction works [18, 15, 21]. We plan to try this idea on our model since it works well on skeleton extraction. Figure 2.4 shows an example of a side-output fusion model. The difference between a side-output model and an ordinary Convolution Neural Network (CNN) is that the former adds additional side-output layers. Side-output layers usually are combinations of a convolutional layer and an upsampling layer that take feature maps in the middle of the CNN as input, and output a skeleton image. Loss functions are usually designed to minimize distances between the side-outputs and the ground truth; the final output is obtained from a fusion of the side-output images, which
Figure 2.4. An example of the side-output mechanism: the model of HED [23]. 3D Boxes represent feature maps in CNN, the resolution of feature maps will decrease during the training, the number of channels usually increased. Side-outputs of each layer will be fused to get final output.

consists of a concatenation of side-output images followed by the application of a final convolutional layer.

The reason for this method to work well on the skeleton is that in each CNN layer, their receptive field scale is different. At shallow CNN layers, the size of convolution kernels is usually too small to detect skeleton features of the thick part of an object; at deep layers, convolution kernels are too big to detect for the thin part of objects. As shown in Figure 2.5, the size of the convolution kernel needs to be larger than the scale of the object to detect the skeleton.
Figure 2.5. The size of the convolution kernels will decide whether a neural network can detect the skeleton. (Figure taken from [18])
CHAPTER 3
CONTRIBUTIONS

Our SSI approach for sketch-to-photo synthesis can be split into the following steps: first, we transform sketches into skeletons; then we use the CycleGAN model to learn a transformation from skeleton to images.

We also tested other methods for the two steps above:

- We process the skeleton by applying either distance transformation or Context Flux, which is then fed into CycleGAN.

- We replace the original model $F$ with the side-output model introduced in Section 2.6. Recall that $F$ is used to transform image to skeleton. In original CycleGAN, $F$ is a series of residual networks (same as $G$), and we would like to see whether side-output model can improve our task performance.

- We design a side-input model to replace the original model for $G$, in an attempt to improve performance.

3.1 Transforming a sketch to a skeleton

To get skeletons from sketches we propose to take the following steps:

- Apply morphology closing operation on sketch images.

- After the closing operation, sketches would become closed contours; then we apply a contour filling algorithm to create masks of sketches.
We use a digital pattern thinning algorithm [26] to skeletonize the mask, finally obtain a skeleton of the sketch. The entire process is shown in Figure 3.1.

This method can remove the distortion of shapes between sketches and objects: as Figure 1.1 shows, the sketch of horse have different shape with real horse; meanwhile, their skeletons are basically same.

3.2 Basic structure: CycleGAN

We are using CycleGAN [27] as our baseline. Our models use the same structure as CycleGAN: there are two generators, $G$ and $F$, where $G$ will translate skeletons $\{S_i\}_{i=1}^{N} \in S$ to images, and $F$ will translate images $\{I_j\}_{j=1}^{N} \in I$ to skeletons. $D_S$ is used for discriminating skeletons $S$ and translated skeletons $F(I)$, and $D_I$ is used for distinguishing between images $I$ and translated images $G(S)$. Like vanilla GAN, CycleGAN also uses adversarial losses to encourage the feasibility of translated images. For our task, the learning object function could be expressed as:

$$
\mathcal{L}_{GAN}(G, D_I, S, I) = \mathbb{E}_{I \sim P_{data(I)}} [\log D_I(I)] + \mathbb{E}_{S \sim P_{data(S)}} [\log(1 - D_I(G(S)))], \quad (3.1)
$$

$$
\mathcal{L}_{GAN}(F, D_S, I, S) = \mathbb{E}_{S \sim P_{data(S)}} [\log D_S(S)] + \mathbb{E}_{I \sim P_{data(I)}} [\log(1 - D_S(F(I)))] , \quad (3.2)
$$
where $G$ translates a skeleton to an image, $D_I$ discriminates whether the input is real images, vice versa for $F$ and $D_S$. Besides the adversarial loss, CycleGAN also has Cycle Consistency Loss:

$$
\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{I \sim P_{\text{data}}(I)}[\|G(F(I)) - I\|_1] + \mathbb{E}_{S \sim P_{\text{data}}(S)}[\|F(G(S)) - S\|_1]. \quad (3.3)
$$

The learning objective function is:

$$
\mathcal{L}(G, F, D_I, D_S) = \mathcal{L}_{\text{GAN}}(G, D_I, S, I) + \mathcal{L}_{\text{GAN}}(F, D_S, I, S) + \lambda \mathcal{L}_{\text{cyc}}(G, F). \quad (3.4)
$$

A sketch that motivates this objective function is shown in Figure 3.2.

**Figure 3.2.** A block diagram that shows CycleGAN’s structure: $G$ translates skeleton to image, meanwhile $F$ translates image to skeleton. Cycle-consistency loss enforces reconstructed results to be similar to the original inputs.
The benefit of CycleGAN is that its training doesn’t require paired data; sketches and images in datasets don’t need to have correspondence to each other. This allows us to collect datasets much more easily. We can extract data from independent sketch and image datasets of a certain class to form our own new dataset.

3.3 Different presentations of skeletons: distance transformation and flux

One characteristic for skeletons is their black background, which occupies the majority of the images. This will cause two issues: 1) for most parts of the skeleton image, the first few convolution layers will have significantly similar outputs; 2) given the small loss variances that would be observed, it would be harder to train the model and tune the hyperparameters. To deal with this problem, we propose to use distance map [3] instead of the skeleton. In a distance map of the skeleton, each pixel’s value is determined by its distance to the skeleton, as shown in Figure 3.3. For the same reason, we also tried to use Context Flux [21] as another kind of representation for skeleton. The Context Flux is described in Section 2.5.

![Figure 3.3](image)
The distance transformation map and the Context Flux can add features on the background of the skeleton; the former adds distance features, while the latter adds direction features. Additionally, the distance transformation map assign every pixel on image with a feature, and all values change gradually, which enriches the information in the whole image. The Context Flux only adds features around the skeleton; compared with the former, it is easier for CNN to detect since it has a clear border, but meanwhile many pixels on the background don’t have meaning.

### 3.4 Side-output model

In Section 2.6 we introduced side-output model, a popular method in skeleton extraction. One of our proposed approaches will replace the original image-to-skeleton generator $F$ with the side-output model. The detail of the model is shown in Figure 2.4.

### 3.5 Side-input model

Inspired by side-output models [23, 18, 15, 21], we propose a novel side-input model for skeleton-to-image translation as shown in Figure 3.4. This model can be seen as a reverse of the side-output model (which is designed for image-to-skeleton translation) introduced in section 2.6, with the expansion of the resolution of feature maps. The skeleton feature will be directly inserted into feature maps and concatenated together.

The benefit of this scheme is that the skeleton pixels can affect different ranges of field on the output image. For example, horse image generation tasks the skeleton pixels of the side-input in the earlier stage will be expanded to form the horse’s torso; in the middle stage the skeleton pixels will be expanded to form its limbs.
Figure 3.4. The building block of the side-input model is upsample and convolution network series. Skeletons are resized to each scale and filled into 1x1 convolution neural network to get side-input features. With forward propagation in the model, side-input features will concatenate with feature maps. (a) represents inserting side-input features into feature maps. (b) represents resizing and 1x1 convolution
CHAPTER 4
SIMULATION RESULTS

4.1 Computation Resources

We are using the UNITY cluster as our computation resource, which can provide plentiful GPU computation. Unity is a High Performance Computing Cluster, for most of our programs, they use 2 of NVIDIA GeForce RTX 2080 Ti from 6 hours to 2 days depends on dataset’s size. For [17] we trained on 4 of NVIDIA GeForce RTX 2080 Ti each up to a week. Note that sometime the resources is not stable, and programs may be stopped for unknown reasons.

4.2 Datasets

We test our methods on six datasets:

1. ShoeV2 [24] has 6,648 sketches (5982 for training, 666 for testing) and 2,000 photos (1800 for training, 200 for testing).

2. ChairV2 [24] has 1,275 sketches (964 for training, 311 for testing) and 400 photos (300 for training, 100 for testing).

3. Zebra class extracted from SketchyCOCO [9], has 2,280 sketches (2091 for training, 189 for testing) and 2,280 photos (2091 for training, 189 for testing).

4. Giraffe class extracted from SketchyCOCO [9], has 2,297 sketches (2065 for training, 232 for testing) and 2,297 photos (2065 for training, 232 for testing).

5. Airplane class extracted from SketchyCOCO [9], has 2,016 sketches (1848 for training, 168 for testing) and 2,016 photos (1848 for training, 168 for testing).
6. Self collected horse dataset: we collected horse images from multiple sources [2, 13, 1], as well as sketches from [14], which have also been cleaned and curated by us: we try to select those images or sketches which contain a horse in full, and flip the images to enrich our data and balance the direction of horses. It has 2,698 sketches (2383 for training, 315 for testing) and 2,208 photos (2043 for training, 330 for testing) before flipping.

4.3 Evaluation Metrics

4.3.1 Frechet Inception Distance

For images generation tasks, the Frechet Inception Distance (FID) is the most commonly used evaluation metric on image synthesis tasks. It can measure the similarity between generated images and real images. To calculate the FID, we need to send generated images $X$ and real images $Y$ into a pretrained Inception V3 model [20] and get their activated outputs from the last pooling layer. Those outputs are activation features of length 2048: we then compute the statistics on the features from $X$ and $Y$, to get their means $\mu_x$ and $\mu_y$ and covariance matrices $\Sigma_x$ and $\Sigma_y$. The FID is defined as follows:

$$FID = ||\mu_x - \mu_y||^2 + \text{Trace}(\Sigma_x + \Sigma_y - 2(\Sigma_x \Sigma_y)^{1/2}).$$ (4.1)

4.3.2 Learned perceptual image patch similarity

Another metric used for images generation tasks is Learned perceptual image patch similarity (LPIPS) [25]. It is based on neural networks trained on a dataset that provides images and pairwise similarity measures; the model’s inputs are 2 images and its output is the distance between them. The training dataset was collected by a two alternative forced choice (2AFC) test, formed by patch triplets $(x, x_0, x_1, h)$, where $x$ is a patch, $x_0$ and $x_1$ are two distortions of $x$; participants were asked which
Figure 4.1. For a patch triplet \((x, x_0, x_1, h)\), first step is computing the distances \(d_0\) and \(d_1\) between the patch \(x\) and the patches \(x_0\) and \(x_1\), respectively; then a small neural network \(G\) will be trained to use \(d_0\) and \(d_1\) to predict perceptual judgment \(h\). After training, the distance computing model on the left will be used for similarity scoring. (Figure taken from [25])

Distortion is closer to \(x\), and record the response \(h \in \{0, 1\}\). How to use patch triples to train the LPIPS model is shown in Figure 4.1.

Similar to [17], we use it to evaluate the diversity of results produced by different methods. LPIPS will get a distance score between each of a group of results, then average them; the averaged distance will show the diversity of this group of images.

4.4 Experimental Results

With the exception of USPS [17], the tested approaches runs for 200 epochs on the same hyperparameters: the learning rate is 0.0002, and will decay to 0 from 100 epoch to 200 epoch; \(\beta_1\) for Adam optimizer is 0.5, \(\beta_2\) is 0.999; the size of the input and output images is 128 \(\times\) 128. We test the FID score every 5 epochs and select the best one for evaluation. LPIPS scores are tested on the epoch that has the best FID score.

From Table 4.1, we can tell that skeleton, distance transform map, flux, side-output model do not show an advantage to CycleGAN, and obviously fall behind USPS [17]. Our approach falls behind USPS might because we didn’t train our models for enough time (due to time and resource availability constraints), and USPS focuses on colorization, while we did not. We didn’t design an independent colorization
Figure 4.2. (a) Sketch. (b) Skeleton. (c) Distance transformation map. (d) Context Flux. Sketch has the most detail and information; for the other three images. The distance transformation map has the fewest sparsity; the skeleton has the largest sparsity.

subsystem due to time and computation resource limits. Our approaches did not show a significant advantage when compare to sketch-to-photo-based CycleGAN, possibly because our skeleton-based method uses images with significant sparsity, as shown in Figure 4.2; the generation task becomes more difficult since for a skeleton more image regions have no meaningful information. Also, even though our SSI approaches do not have advantages on generation quality for all sketches, the results of SSI still can correct the shape of distorted sketches, as shown in Figure 4.3. Among our SSI approaches, the distance transformation-based methods (including the side-output+DT) show the best performance; it might be because these methods reduce image sparsity more than others.

Table 4.2 shows the diversity score for each of the approaches. The LPIPS does not show a clear winner among the approaches. We can see that skeleton and Context Flux sometime create pretty bad scores; this might caused by their large sparsity.

The self-designed side-input model trends worse than other methods. We conjecture that a novel model design requires a significant fine tuning of its parameters; another possibly reason is some side-inputs are too close to the output, which means their features are not ready for realistic image generation. Given its poor performance, seen in Figures 4.4 and 4.5, we limit its application to a handful of datasets.
Figure 4.3. Examples of images generated by each method. We can see that for most cases, there is no significant difference between those methods. For sketches that have huge distortions, like column 4 and 6, SSI and SSI+Distance transformations can produce better shapes. SSI+Flux didn’t produce realistic images, possibly caused by randomness in neural network training.
### Table 4.1. FID score for different methods. Note that lower FID score means more realistic results. DT meas distance transformation. “side-output” means use side-output model replace the original residual networks. “*” denotes missing data. (*) denotes results form incomplete training (the only case in the table is trained for 195 epochs due to an unknown cluster issue), then program crashed due to computation resources limit. Note that bold numbers are the best scores among all methods in a dataset. Bold italic numbers are the best score for our proposed methods.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ShoeV2</th>
<th>ChairV2</th>
<th>horse</th>
<th>zebra</th>
<th>giraffe</th>
<th>airplane</th>
</tr>
</thead>
<tbody>
<tr>
<td>USPS[17]</td>
<td>48.73</td>
<td>100.51</td>
<td>99.45</td>
<td>33.02</td>
<td>139.08</td>
<td>113.49</td>
</tr>
<tr>
<td>CycleGAN[27]</td>
<td>74.76</td>
<td>124.96</td>
<td>139.74</td>
<td>35.57</td>
<td>145.72</td>
<td>127.54</td>
</tr>
<tr>
<td>SSI</td>
<td>77.11</td>
<td>178.19</td>
<td>191.42</td>
<td>52.51</td>
<td>147.14</td>
<td>158.1</td>
</tr>
<tr>
<td>SSI+DT</td>
<td>63.94</td>
<td>145.86</td>
<td>123.18</td>
<td>46.86</td>
<td>144.12</td>
<td>129.31</td>
</tr>
<tr>
<td>SSI+Flux</td>
<td>*</td>
<td>173.23</td>
<td>122.55(*)</td>
<td>56.66</td>
<td>150.95</td>
<td>188.14</td>
</tr>
<tr>
<td>SSI+side-output</td>
<td>70.63</td>
<td>179.09</td>
<td>169.93</td>
<td>53.12</td>
<td>146.87</td>
<td>125.28</td>
</tr>
<tr>
<td>SSI+side-output+DT</td>
<td>70.84</td>
<td>170.9</td>
<td>112.78</td>
<td>40.61</td>
<td>144.12</td>
<td>164.43</td>
</tr>
<tr>
<td>SSI+side-input+DT</td>
<td>*</td>
<td>*</td>
<td>169.67</td>
<td>*</td>
<td>171.21</td>
<td>*</td>
</tr>
</tbody>
</table>

### Table 4.2. LPIPS score for different methods correspond to 4.1. Note that higher LPIPS score means better diversity. “*” denotes missing data, (*) denotes results form incomplete training. Note that bold numbers are the best scores among all methods in a dataset. Bold italic numbers are the best score for our proposed methods.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ShoeV2</th>
<th>ChairV2</th>
<th>horse</th>
<th>zebra</th>
<th>giraffe</th>
<th>airplane</th>
</tr>
</thead>
<tbody>
<tr>
<td>USPS[17]</td>
<td>0.146</td>
<td>0.156</td>
<td>0.397</td>
<td>0.455</td>
<td>0.432</td>
<td></td>
</tr>
<tr>
<td>CycleGAN[27]</td>
<td>0.238</td>
<td>0.403</td>
<td>0.41</td>
<td>0.479</td>
<td>0.48</td>
<td>0.424</td>
</tr>
<tr>
<td>SSI</td>
<td>0.293</td>
<td>0.037</td>
<td>0.314</td>
<td>0.014</td>
<td>0.479</td>
<td>0.246</td>
</tr>
<tr>
<td>SSI+DT</td>
<td>0.328</td>
<td>0.338</td>
<td>0.442</td>
<td>0.464</td>
<td>0.49</td>
<td>0.417</td>
</tr>
<tr>
<td>SSI+Flux</td>
<td>*</td>
<td>0.23</td>
<td>0.458(*)</td>
<td>0.052</td>
<td>0.469</td>
<td>0.173</td>
</tr>
<tr>
<td>SSI+side-output</td>
<td>0.307</td>
<td>0.19</td>
<td>0.297</td>
<td>0.016</td>
<td>0.49</td>
<td>0.424</td>
</tr>
<tr>
<td>SSI+side-output+DT</td>
<td>0.307</td>
<td>0.297</td>
<td>0.304</td>
<td>0.407</td>
<td>0.476</td>
<td>0.24</td>
</tr>
<tr>
<td>SSI+side-input+DT</td>
<td>*</td>
<td>*</td>
<td>0.304</td>
<td>*</td>
<td>0.388</td>
<td>*</td>
</tr>
</tbody>
</table>
Figure 4.4. CycleGAN and SSI methods tested on giraffe dataset. We can see that with further model training, the FID scores are decreasing, and most methods exhibit comparable performance, except for side-input models which have worse performance.

Figure 4.5. CycleGAN and SSI methods tested on horse dataset. We can find for many cases the methods that performed well with the giraffe dataset did worse here, and side-input models have bad performance almost throughout the training process.

Figure 4.4 and Figure 4.5 show all of our proposed methods and CycleGAN’s FID score as a function of the training epoch on the giraffe and horse datasets. We can see that the SSI and SSI+side-output approaches do not perform well on horse
datasets; since these two datasets have no essential differences, we can tell that these two methods are not stable. Also, we can see that self-designed side-input models always have bad performance.

USPS [17] is the most advanced method in sketch-to-photo synthesis. The default setting of it trained for 700 epochs; we record their performance with intervals of 5 epochs, and compared with other methods in Figure 4.7. We can see that in the first 200 epochs, USPS does not show a significant advantage over other methods. The reason we didn’t test 700 epochs for our approaches is mainly because there is no enough time and resources for large scale testing. A second reason is that 200 epochs is the default setting of CycleGAN; we want a fair condition for our baseline. We can only guess that if we increase our training epochs, the performance would become better: we can see that most methods are almost getting convergence at 200 epochs. USPS didn’t change too much after 200 epochs (except for horse dataset, since horse data is more complex, which means harder to generate); but even we got to the convergence point, more training time still can get better FID score because the possibility to reach lower points is also increased.
Figure 4.6. CycleGAN and SSI methods tested on ShoeV2, horse and zebra. These three panels show that no single method will win on all datasets. However, the skeleton-based method tends to be worse than others.
Figure 4.7. USPS compared with CycleGAN and SSI methods. In this figure, red lines are USPS’s FID scores. Note that we do not show some methods which have very large FID scores to make the figure easy to view.
CHAPTER 5
CONCLUSIONS

To deal with distortion problems in sketches, we proposed Sketch-to-Skeleton-to-Image (SSI) method. We also try to use distance transformation (DT) and Context Flux as additional representations of skeletons. For image-to-skeleton translation we tried side-output model, which is a popular method for skeleton extraction; we also designed a side-input model for skeleton-to-image translation.

We tested the methods above on six different datasets, and compared with CycleGAN, we saw similar performances among methods on every dataset. Although some of the proposed methods perform better than CycleGAN on specific datasets, we cannot claim consistent improvements due to fluctuations in deep learning training. One possible reason for this behavior is that the hyperparameters we used are designed for original CycleGAN, we might be able to achieve better performance if we try different groups of hyperparameters for our methods, as the images we use are much more sparse. Another reason for SSI not to outperform CycleGAN might be the essential drawback of skeletons: it is more sparse than a sketch, which makes image-to-image translation harder.

From Table 4.1, we are able to say USPS [17] has the best performance. Part of the reason is that we trained it for a much longer time, which makes it more likely to achieve a better score.

FID results on our approaches show that the distance transformation map is a better skeleton representation; LPIPS shows that skeleton-based and Context Flux-based methods are not stable, since sometimes have extremely low performance. All
these clues imply the sparsity of input data has a negative correlation with the quality of the generated images.

We can further analyze how different datasets affect the FID score; note that FID score is more important since it can represent the quality of generated images. If we compare our best results with the USPS, we will find that although on all datasets we fall behind them, for horse, zebra, giraffe and airplane datasets, our FID results are closer to theirs. These four kinds of objects have a standard correct shape, therefore skeletonization schemes have better performance. In contrast, shoes and chairs have no single correct shape, and the gap between ours and USPS is larger.
BIBLIOGRAPHY


