Perceptual-Aware Sketch Simplification Based on Integrated VGG Layers

Xuemiao Xu, Minshan Xie, Peiqi Miao, Wei Qu, Wenpeng Xiao, Huaidong Zhang, Xueting Liu, and Tien-Tsin Wong

Abstract—Deep learning has been recently demonstrated as an effective tool for raster-based sketch simplification. Nevertheless, it remains challenging to simplify extremely rough sketches. We found that a simplification network trained with a simple loss, such as pixel loss or discriminator loss, may fail to retain the semantically meaningful details when simplifying a very sketchy and complicated drawing. In this paper, we show that, with a well-designed multi-layer perceptual loss, we are able to obtain aesthetic and neat simplification results preserving semantically important global structures as well as fine details without blurriness and excessive emphasis on local structures. To do so, we design a multi-layer discriminator by fusing all VGG feature layers to differentiate sketches and clean lines. The weights used in layer fusing are automatically learned via an intelligent adjustment mechanism. Furthermore, to evaluate our method, we compare our method to state-of-the-art methods through multiple experiments, including visual comparison and intensive user study.

Index Terms—Convolutional neural network, perceptual awareness, sketch simplification

1 Introduction

Sketching is the foremost step in drawing, allowing artists to quickly draft and visualize their minds. However, the initial works are naturally rough, since they involve several sketchy strokes for presenting one long curve. Therefore, artists will then refine the sketches by converting several overlapped strokes into one clean line and erasing redundant strokes. A tedious and labor-intensive cleaning up or simplification process is usually followed to yield a neat drawing. To automate this simplification process, several methods have been proposed. Early works deal with vector input [1], [2], [3] due to the complication in handling raster input. There are attempts to convert a raster drawing to a vector form [4], but it could be very unstable due to the noises and lighting.

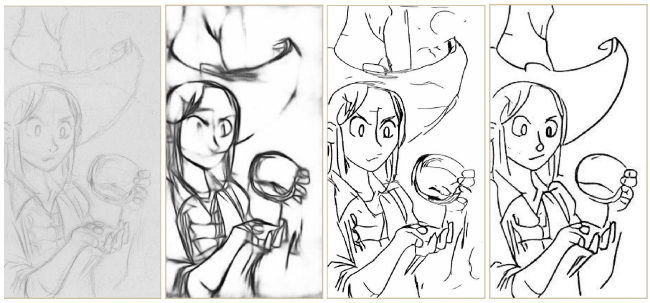
With the advancement of deep convolutional neural networks (DCNN), direct simplification on raster sketch has been effectively demonstrated by Simo-Serra et al. [5], [6]. However, the original DCNN-based sketch simplification [5] is prone to blurriness as it tends to pixel-wisely average the candidate solutions (Fig. 1b). Although vectorization is further applied as post-processing for removing the blurriness, it easily leads to clutter on complex structures and losses its original semantic meaning, like the hand structure in Fig. 2b. Adopting generative adversarial network (GAN) [6] can better avoid the blurriness, as the discriminator network randomly picks from the candidate solutions [7], instead of averaging them. However, it may still fail to simplify extremely sketchy drawing as demonstrated by the bowl in Fig. 1c and the hand in Fig. 2c. This can be explained by the fact that the discriminator network classifies the generated output as a simplified sketch or not, with less attention to the overall semantic structure. As a result, their method tends to over-emphasize on the local structure and preserves many strokes that are less semantically important in a global scale.

To handle extremely sketchy strokes, a more global semantic understanding of the sketch is needed. Recently, the perceptual loss, defined on perceptual features extracted by a pre-trained DCNN (VGG16/VGG19) [8], has been proved to be useful in various applications that need high-level understanding, such as style transfer [7], [9], image inpainting [10], CT image denoising [11], and image synthesis [12]. We found that the perceptual loss is extremely suitable for our sketch simplification task. However, selecting appropriate feature layers is still quite challenging. With low-level features, small-scale details may be better preserved, but large-scale sketchy strokes cannot be well simplified and the results may become distorted as the lack of high-level understanding of semantic structures (Fig. 3b). In contrary, with high-level features, the global semantic structure can be revealed and hence large-scale sketchy strokes can be handled, but small-scale details may be lost (Fig. 3c).

In this paper, we propose a new CNN model for the sketch simplification utilizing the integration of VGG layers. Instead of using only one manually selected layer from the extracted

- *X. Xu is with the School of Computer Science and Engineering, South China University of Technology, Guangdong 510641, China, and also with State Key Laboratory of Subtropical Building Science and Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information, China. E-mail: xuemx@scut.edu.cn.*
- *P. Miao, W. Qu, W. Xiao, and H. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangdong 510641, China. E-mail: {201621032177, csvivianqu, 201630599500, z.huaidong}@mail.scut.edu.cn.*
- *M. Xie and T.-T. Wong are with the Chinese University of Hong Kong, China. E-mail: {msxie, ttwong}@cse.cuhk.edu.hk.*
- *X. Liu is with the School of Computing and Information Sciences, Caritas Institute of Higher Education, China. E-mail: tliu@cihe.edu.hk.*

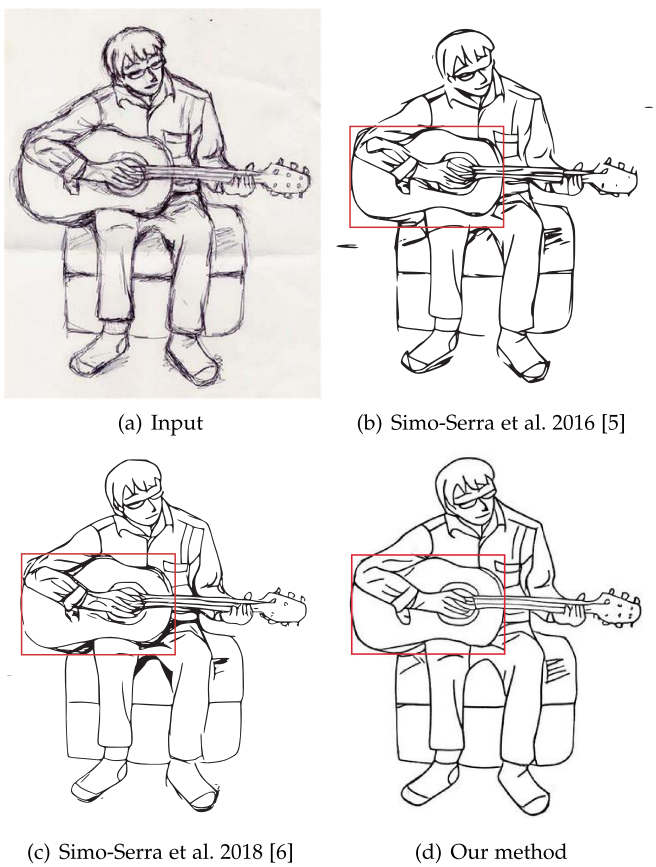
Manuscript received 19 Mar. 2019; revised 15 June 2019; accepted 17 July 2019. Date of publication 24 July 2019; date of current version 24 Nov. 2020. (Corresponding author: Xuemiao Xu.)
Recommended for acceptance by L. Liu.
Digital Object Identifier no. 10.1109/TVCG.2019.2930512



(a) Input (b) Pix. (c) Pix.+GAN (d) Our loss

Fig. 1. (a) Input. (b) Result without vectorization generated by Simo-Serra et al. [5]). Blurry artifacts can be observed. (c) Result without vectorization generated by Simo-Serra et al. [6]. The result is still sketchy. (d) Our result.

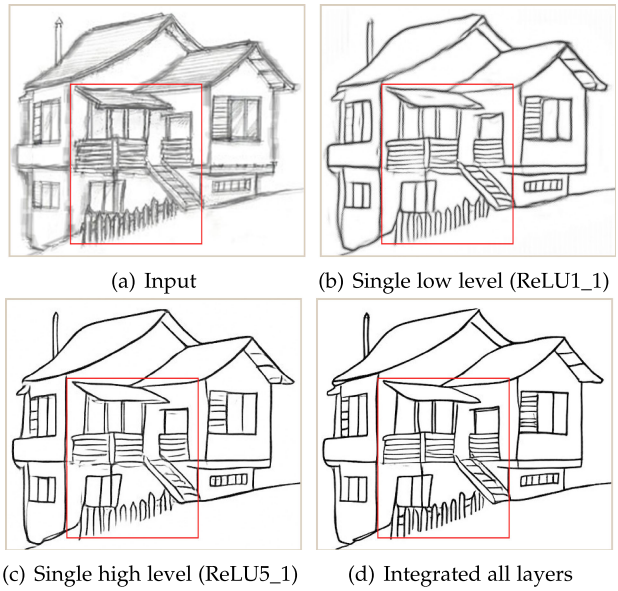
VGG feature layers, we propose to use all feature layers to handle strokes of different scales. Since different scales of sketchiness may distribute in different layers of feature maps, the weights used for integrating the feature layers should vary across different layers. Inspired by some previous works (e.g., saliency detection [13] and semantic segmentation [14], [15]) where all feature layers are involved and automatically combined as the guidance for some specific tasks, we propose to design a multi-layer discriminator by fusing multiple feature layers for our own purpose, i.e., differentiating sketches and clean lines. To balance the importance of different feature layers, we also need an intelligent adjustment mechanism to learn the weights of the layers. The idea is similar to the early Hypercolumn method used in [16], [17]. For every feature map, we stack on an additional convolutional layer. This additional convolution layer will be used to adjust the weight of the corresponding feature layer. Although this approach looks quite straightforward, we found that it works quite well in learning the optimal weights for our purpose. Overall, our method consists of two losses: the perceptual loss which guarantees the visual similarity between the output clean line drawing by our network and the ground truth labelled by users, and the discriminator loss which guarantees that the output clean line drawing looks real.



(a) Input (b) Simo-Serra et al. 2016 [5]

(c) Simo-Serra et al. 2018 [6] (d) Our method

Fig. 2. Comparison of our method to state-of-the-art methods. Note that vectorization is applied as post processing for [5] and [6] to remove the blurriness, while our results is directly outputted by our simplification network. Note how our method suppresses the blurriness and well simplifies extremely sketchy drawing faithfully to its semantic meaning.



(a) Input (b) Single low level (ReLU1_1)

(c) Single high level (ReLU5_1) (d) Integrated all layers

Fig. 3. (a) Input. (b) Result with perceptual loss only relied on a single low-level feature layer (ReLU1_1). (c) Result with perceptual loss only relies on a single high-level feature layer (ReLU5_1). (d) Our result.

To validate the effectiveness of our method, we compare our method to existing methods qualitatively and by the user study due to the lack of quantitative measurement. Our method outperforms the state-of-the-art methods in all aspects, including overall aesthetics, tidiness of lines, and content conformity to the input. Our contributions can be summarized as follows:

- We provide a deep neural network tailored for sketch simplification in which a multi-layer discriminator is designed to guide the combination of all VGG feature layers.
- We propose an adaptively learned perceptual loss which can be aware of more global semantic information and this is achieved by utilizing an adjustment mechanism to automatically learn the weights for better balancing the importance of different feature layers.
- We construct a new, rich and challenge rough sketch dataset existing large-scale strokes and important details simultaneously, and containing scanned images and camera-catched images with different illumination. Our dataset aims to be more approximate to real application scenarios by applying bi-directional construction. Therefore we can cover many different types of drafts.

2 Related Works

2.1 Sketch Simplification

Sketch simplification is a classical problem that has been studied by many researchers. Among the existing works, most are worked on vector input. To help artists retrace the sketchy strokes efficiently, various interactive systems have been proposed [18], [19]. However, these systems only accept vector input, and tedious manual effort is still needed for complex sketches. To automatically convert sketchy strokes into clean line drawings, methods have been proposed to simplify the sketch by removing insignificant strokes based on 2D stroke properties such as position, density, and length [20]. For strokes that are obtained from 3D models, the depth and silhouette information can also be used for measuring the significance of strokes [1], [21]. While these stroke reduction methods only work on vector input, they are also unable to obtain a long and smooth curve from a set of short sketchy strokes. The stroke grouping methods are the state-of-the-art sketch simplification methods for vector input [2], [3], [22]. These methods generally group the sketchy strokes into stroke groups based on inter-stroke proximity, continuity, parallelism, and closure. Each stroke group is then replaced by a single smooth curve. Similar to above, the stroke grouping methods still only accept vector input.

For raster sketches, methods have been proposed by first extracting line segments from the image and then connecting the line segments into smooth curves [4], [23], [24]. However, these methods are extremely unstable, especially when the strokes are quite rough, or the raster sketch contains the noise/shadow due to imperfect scanning or photo-shooting. Recently, deep convolutional neural network based approaches have shown much better performance towards sketch simplification. In particular, Simo-Serra et al. [5] proposed a fully-convolutional network to simplify raster sketches by minimizing the MSE loss. However, as we have discussed in Section 1, the MSE loss generally leads to blurry artifacts in the generated clean line drawing. To tackle the blurring issue, Simo-Serra et al. [6] further proposed to employ the generative adversarial networks (GANs) and incorporated the adversarial loss into the loss function. While this approach may tackle the blurring issue, it still cannot well handle extremely sketchy strokes since GAN only guarantees local cleanness while paying less attention to the overall semantic structure. In contrast, by adopting the multi-layer perceptual loss, our method well handles extremely sketchy strokes with different scales. Besides, the training dataset used by existing networks only contains clean sketches and fails to generate satisfying results for real-world sketches with uncontrolled lighting conditions. We resolve this issue by augmenting the training sketch examples with different modifications, for example, introducing different lighting conditions to the input sketch. Experiments show that our method well handles all types of real-world sketches and greatly outperforms the existing methods.

2.2 Perceptual Loss in DCNN

Recent works have shown that adopting pixel loss (e.g., pixel-wise MSE) in image-to-image DCNN generally leads to over-smoothing and lacking details [7], [9]. Therefore, the perceptual loss is frequently needed to measure the difference between output and ground-truth. In particular, the VGG16 network [8] is proposed to extract high-level perceptual features and is widely used in tasks where high-level semantic differences are preferred, such as style transfer and super resolution [7], [9], inpainting [10], [25], image synthesis [12], and denoising [11]. Here, the perceptual loss is generally formulated as the loss of one or several specific feature layers in a VGG network, and the choice is usually application-dependent. For example, a high-level feature layer (ReLU5_4) is chosen for [9], [11], while a low-level feature layer (ReLU2_2) is chosen for [12]. Two feature layers (ReLU3_1 and ReLU4_1) are chosen in [25]. In this paper, instead of relying on one or multiple selected layers, we leverage the information from all layers via automatic learning and calculate the perceptual loss based on all feature layers.

2.3 Feature Integration in FCNs

Feature layers are frequently used as the higher-level description of the input image and serve as the base for various applications. Since multiple feature layers exist at the same time, methods have been proposed to combine multiple feature layers into one guidance map towards different applications, such as segmentation [26], localization [16], image perceptual similarity [27] and saliency detection [13]. To combine multiple feature layers into one, some methods proposed to link low-level and mid-level features to prediction layers directly through concatenating layers [14], [15]. Hariharan et al. [16] proposed the hypercolumn representation to equivalently extract per-pixel descriptors for fine-grained localization tasks. Larsson et al. [17] proposed to train their system for the task of predicting hue and chroma distributions for each pixel given its hypercolumn descriptor. Similarly, Ronneberger et al. [26] proposed to adopt multiple skip-connections to build a contracting path to capture context and a symmetric expanding path that assembles more precise segmentation in their Unet. Moreover, Zhang et al. [13] proposed to integrate all level features into multiple resolutions at once so that coarse semantics and fine details are incorporated simultaneously to generate the nal saliency maps. Our work shares the same spirit but tailors for our application. In particular, we design a multi-layer discriminator that automatically fuses feature layers to differentiate sketches and clean lines.

3 Overview

Our network is illustrated in Fig. 4, which consists of two parts, a simplification network and a multi-layer discriminator network. The simplification network takes an arbitrary raster sketch image as input and generates a simplified clean line drawing as output. The detailed architecture of the simplification network is presented in Section 5.1. Our multi-layer discriminator network is designed to generate the adjusted feature maps, which can well balance the importance of different layers and channels automatically. The generated feature maps are further used for calculating the multi-layer perceptual loss. The discriminator takes all 13 feature layers extracted by the fine-tuned VGG16 model (in pink) as input. We fine-tune the VGG16 to capture the semantics of sketches and line drawings. To do so, we borrow the idea of GoogleNet [28] to fine-tune the pre-trained model of VGG16 with batch normalization [7]. To integrate

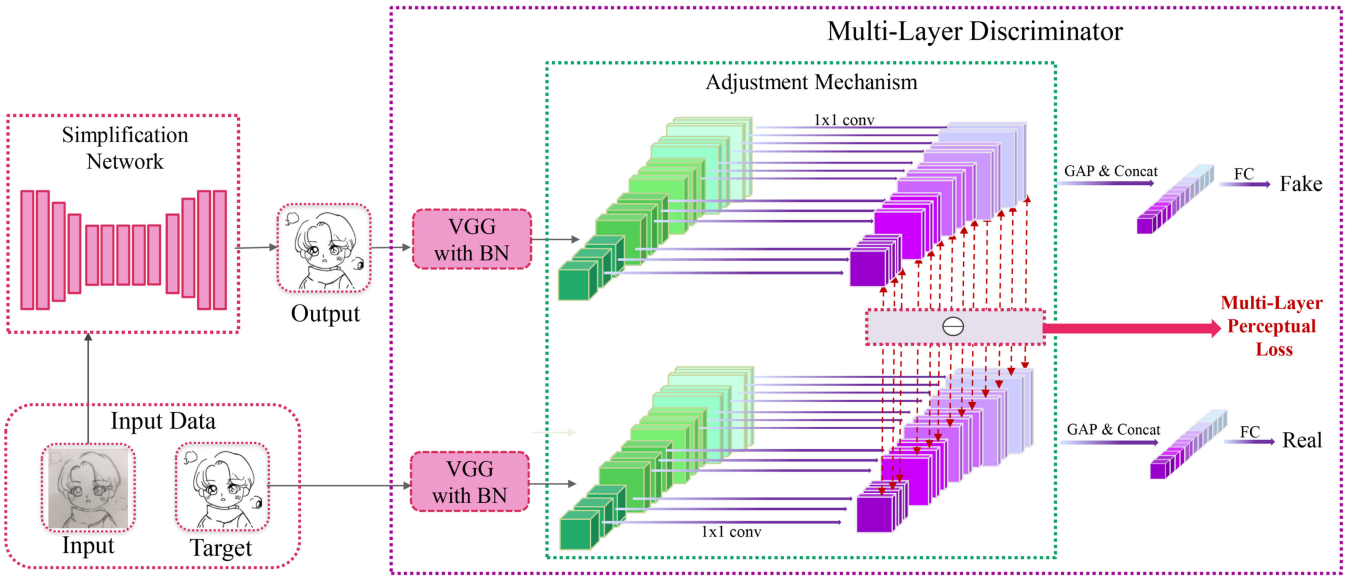


Fig. 4. System overview. We extract all 13 layers from VGG model (in green) and do the adjustment mechanism guided by our multi-layer discriminator. The multi-level perceptual loss is measured over the features after adjustment and will be backpropagated as the training loss of simplification network (in red).

different VGG layers, an adjustment mechanism is applied so that the weights used in layer integration can be automatically learned. Finally, our multi-layer perceptual loss can be calculated based on the adjusted feature layers (presented as gradually varied purple blocks). The detailed design of the multi-layer perceptual loss is presented in Section 4.1.

4 Method

4.1 Adaptively Learned Perceptual Loss

Before introducing our adaptively learned perceptual loss, we first give a brief introduction of the existing single-layer based perceptual loss [7]. In all, the perceptual loss measures the semantic difference between two input images at the feature level. Comparing to training with losses at image level (such as MAE and MSE), training with the perceptual loss tends to generate more semantic-sensitive samples with less blurring artifacts. Since our sketch simplification task needs a high-level understanding of both the sketch and the clean line drawing, we find that the perceptual loss is extremely useful in preserving the semantic structure between sketches and clean line drawings. The perceptual loss between an output $\hat{I}_y$ and the ground truth I_y for the ith feature layer [9], [12], [25] is defined as

$$\mathcal{L}_{VGGs}(\hat{I}_y, I_y) = \frac{1}{C_i H_i W_i} \|\phi_i(\hat{I}_y) - \phi_i(I_y)\|_2^2. \quad (1)$$

Here, $\phi_i(x)$ is the function that obtains the feature map for x by the ith convolution (after activation) with our fine-tuned VGG model. W_i, H_i and C_i denote width, height, and channel size of $\phi_i(x)$ respectively. By defining the perceptual loss on a single feature layer, we can achieve relatively good result (Figs. 3b and 3c). However, as we have stated, with a single high-level feature layer, large-scale sketchy strokes can be handled, but small-scale details may be lost. In the contrary, with a single low-level feature layer, small-scale details may be better preserved, but large-scale sketchy strokes may not be well simplified. Therefore, we propose to integrate all feature layers by learning the optimal weights.

In the following subsections, we illustrate our feature integration by an adjustment mechanism with the guidance of our multi-layer discriminator.

4.1.1 *Multi-Layer Discriminator*

In order to fully exploit the complementary information scattered among multiple layers of features, we introduce a multi-layer discriminator which aims to distinguish sketches and clean line drawings, to guide the integration of all features by learning. By integrating all feature layers, the discriminator would be able to recognize the distinctive features of all feature layers, and therefore can better discriminate sketches and clean line drawings.

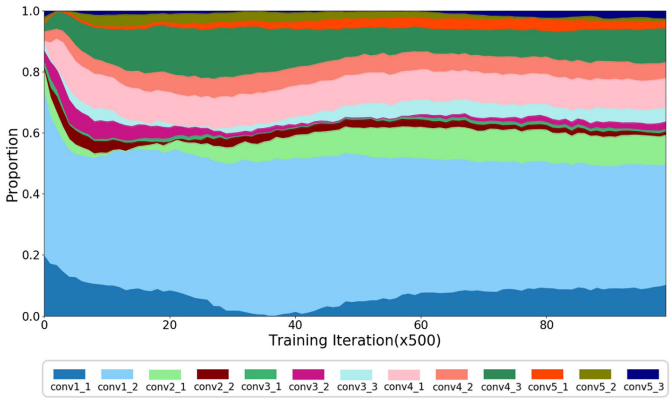
Our discriminator takes all 13 layers of features extracted by our fine-tuned VGG model as the input. These feature maps are then adjusted through the following mechanism. For each feature layer $\phi_i(x)$ extracted by the VGG, we denote its corresponding adjusted feature layer as $\phi_i'(x)$. For each adjusted feature layer $\phi_i'(x)$, we further apply a global average pooling layer (GAP) and concatenate the vector results followed by a fully connected layer (FC). Finally, the discriminator will predict a single value of P to judge whether the input is a sketch or a clean line drawing.

$$P(x) = FC(GAP(\phi_i'(x))). \quad (2)$$

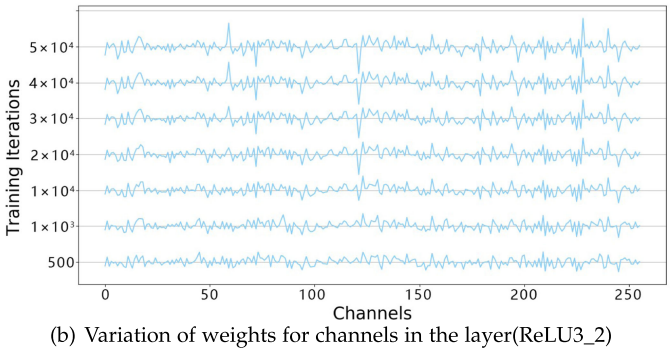
The discriminator loss follows the principle of GAN [29] and is similarly defined as

$$\mathcal{L}_D(\hat{I}_y, I_y) = -log(P(I_y)) - log(1 - P(\hat{I}_y)). \quad (3)$$

With this multi-layer discriminator design, the feature layers can be effectively integrated so as to distinguish sketches and clean line drawings better.



(a) Variation of weights for layers



(b) Variation of weights for channels in the layer(ReLU3_2)

Fig. 5. Visualization of the variation of weights during adjustment process.

4.1.2 Adjustment Mechanism

During the layer integration, we intend to set higher weights for layers containing more sketch features in order to detect and further smooth the sketchy regions. The idea is similar to the existing approaches [16], [27]. We add one convolutional layer for each feature layer for automatically learning the layers weight. That is, for each feature layer $\phi_i(x)$, we introduce a 1×1 convolutional layer to adjust its channel size to 64. This adjustment layer is trained by the discriminator loss together with the perceptual loss, so the adjusted weights are able to effectively balance the feature layers in terms of sketchiness.

To validate the effectiveness of the 1×1 convolutional layer used for feature layer adjustment, we visualize the learned weights during the adjustment process in two forms. First, we calculate the weight of each feature layer by averaging all weights of the channels in the corresponding 1×1 convolutional layer. Fig. 5a plots the result. After a couple of iterations, the weights of the feature layers are more or less stabilized. We also experiment with different initial values. Similar results are obtained. We can observe from the figure that lower level layers are of higher weights, as they capture more sketch information. We further visualize the weights of each channel for each feature layer. Fig. 5b visualizes the result. We can see that the weights of the channels are also stabilized during the adjustment process (from the bottom curve to the top stabilized curve).

4.1.3 The Loss Design

Based on these adjusted feature layers, we define the multi-level perceptual loss as the integrated difference between the output $\hat{I}_y$ and the ground truth I_y over all feature layers

$$\mathcal{L}_{VGGa}(\hat{I}_y, I_y) = \sum_{i=1}^{n} \frac{1}{C_i' H_i' W_i'} \|\phi_i'(\hat{I}_y) - \phi_i'(I_y)\|_2^2. \quad (4)$$

Here W_i', H_i' and C_i' are the width, height, and channel size of $\phi_i'(x)$ respectively. Note that the weights are implicitly embedded in the 1×1 convolutional layer used for adjustment.

The discriminator loss is used to further guarantee the generated clean drawing looks like a real clean line drawing. Then we combine the multi-layer perceptual loss and discriminator loss as our total loss for training the simplification network

$$\mathcal{L}_G(\hat{I}_y, I_y) = \mathcal{L}_{VGGa}(\hat{I}_y, I_y) + log(1 - P(\hat{I}_y)). \quad (5)$$

The simplification network is trained by minimizing the loss function. Comparing to training on a single feature that only captures a specific scale of sketchiness, training on all feature layers can capture sketches with different scales of sketchiness (Fig. 3d).

4.2 Rough Sketch Dataset

In order to simulate reality application scenarios more accurately, we construct a new rough sketch dataset, which holds various and challenging types. The dataset contains 140 well-labelled sketches for training and 21 unlabelled sketches for user evaluation. The images are collected under different resolution, illumination, roughness and drawing styles. Due to the lack of the paired sketches and its clean line drawings, drawing from the human to form the pairs are needed, which is very labor-intensive and time-consuming. To prevent the personal drawing habits of artists leading to potential hints in the line drawing, 10 artists are asked to draw the clean line drawings. Specifically, the total 140 sketches are allocated into 10 different sketches groups, and each artist is responsible for his/her group. The drawing software is Adobe Illustrator, which can produce lines in vector form, so that the images can be zoomed in and out without blurriness and distortion.

With regard to the inversed dataset construction approach employed by [5], they asked artists to trace the draft based on the given clean line drawing. As shown in Fig. 6a, a drawback of this approach is that the constructed sketch line is too simple and limited as the sketches drawn by artists have little deviation when the positions of initial lines are given. Furthermore, this type of line may be inconsistent with the lines of the real world. In order to ensure our dataset capable of handling various kinds of sketches, we bi-directionally construct the new rough sketch dataset. Specifically, the pair-dataset with direct construction and inverse construction are both applied in our dataset preparation. It means that the artists are asked to draw the clean line drawing (Fig. 6b), or do the inverse. In this bi-directionally way, the constructed pair data can better simulate the reality scenarios because artists generally use multiple strokes to present one curve.

To include various application scenarios, we collect rough sketches of varied objects, e.g., portraits, plants, animals,

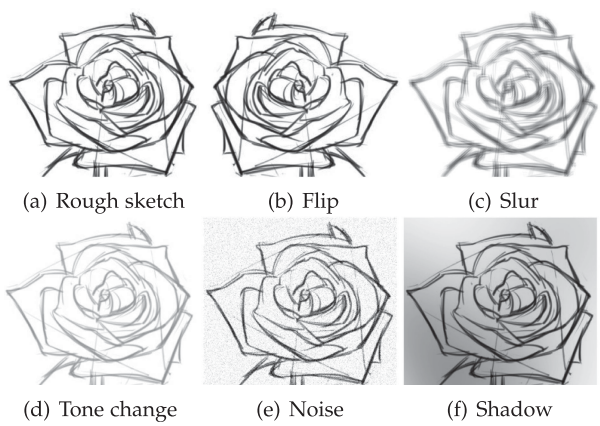
(a) Inversed construction (b) Direct construction (c) Large strokes & details (d) Scanned image (e) Camera-catched image

Fig. 6. Here we show parts of the dataset we collected where the first line is rough sketches and the second line is the corresponding clean line drawings. For purpose of getting closer to real scenes, we not only follow the inversed construction used by [5]'s (Fig. 6[a]) but also apply the direct way to generate clean line drawings. Meanwhile, our dataset contains many challenging types, such as scanned images (Fig. 6[d]) and camera-catched images (Fig. 6[e]) with uncontrolled light condition, which were not handled by previous methods.The images cover a variety of drawing types, e.g., portraits, buildings, objects, cartoon; The roughness of sketches is also in vast variance. In order to make our dataset more robust to uncontrolled light condition in real world, we have collected examples of uncontrolled light condition.

buildings, industrial design, and it also covers sketches drawn by pens, ballpoint pens, pencils, and digital drawing, etc. Some of the sketches collected were scanned (Fig. 6d) or camera-catched (Fig. 6e) with uncontrolled illumination which are quite difficult to obtain pretty simplified results. To better solve the problem of removing large-scale strokes and preserving details at the same time, our data set generally contains both cases. For example, refer to Fig. 6c, the collar on the dinosaur's neck (in the red box) can be represented by one line in our perception, but a lot of strokes used here; the horns on the back and teeth are fine-detailed areas shown in the blue box, although the lines are very close and dense, these are semantically meaningful parts and should not be removed as miscellaneous lines.

Except for the collected dataset, we apply well-designed data augmentation methods to have more times the images. We perform these data augmentation approaches: scaling, flipping, slurring, tone-change, adding noise and shading. To train on the roughness of sketches varying from different scales, after getting the training line drawings and sketches, we zoom in on the image at different scales, $1.0, 1.2, 1.5, 1.8, 2.0$ respectively. In addition, sketch is flipped horizontally (Fig. 7b), vertically, and diagonally. Image slur is done by using the method proposed by [5], and this can be seen in Fig. 7c. To tolerate sketches with varying tones, noises, and shading, we further randomly change the tone (Fig. 7d), add noise [5] (Fig. 7e), and add shading to each sketch (Fig. 7f), each with 2 sets of different augmentation parameters.

After augmentation, $140 \times 5 \times 4 \times 2 \times 2 \times 2 \times 2 = 44800$ sketches are obtained. Note that the same line drawing is used as the output for all sketches augmented from the same original sketch. With the collected and augmented sketches, we randomly collect 384×384 resolution patches from the sketches as the final training examples. $N_I = floor(W_I H_I / 384 \times 384)$ patches are collected from each sketch I. In total, we collect 48896 patches.



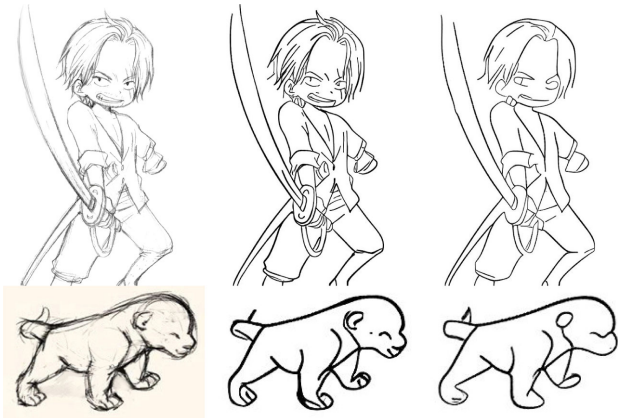
(a) Rough sketch (b) Flip (c) Slur (d) Tone change (e) Noise (f) Shadow

Fig. 7. Samples applying data augmentation.

5 Results and Discussion

To validate the effectiveness of our method, we apply our method on a large set of sketches with different types, including both carefully scanned and casually photographed sketches. There are sketches drawn with pencils, pens, or ball pens. We do the ablation analysis of our method, and further compare our method to two state-of-the-art CNN-based methods [5], [6]. For a fair comparison, we also fine-tuned and retrained their models with our training data.

To validate the effectiveness of our multi-layer scheme, we also compare the results generated by perceptual loss on a single layer (*SL*), perceptual loss on the average of all layers (*AL*), and perceptual loss on all layers with automatically learned weights (*AM*). Note that we use the feature layer that produces the optimal result (lowest loss) as the single layer approach. As suggested by user study (Section 5.3), we select the ReLU5_1 layer as the optimal single layer.



(a) Input (b) Using 1×1 (c) Using 3×3

Fig. 8. Ablation study on the effectiveness by using 1×1 convolutional layer of the adjustment mechanism.

5.1 Simplification Network Details

We implement our simplification network based on FCN architecture [15] and follow the architecture guidelines for the image transformation network in [9], [30]. Specifically, we employ 13 convolutional layers with the same 3×3 kernel size (4×4 kernel at the $10th-12th$ layer). The first 12 layers are followed by batch normalization (BN) and ReLU. To ensure better semantically understanding of complicated structures, we enlarge the receptive field by applying convolution layer and deconvolution layer with stride 2 at $2-4th$ and $10-12$th layer respectively. In this way, the receptive field is effectively enlarged without losing too much sketch details. For better performance and faster convergence, we implement the $5th-9th$ convolution layers as 5 residual units [31] with 256-dimensional input and output feature maps.

5.2 Visual Comparison

5.2.1 Ablation Analysis

Varying Convolutional Layers. To validate the effectiveness of the 1×1 convolutional layer, we compare the results generated by 1×1 convolutional layer and 3×3 convolutional layer respectively for feature adjustment. The result is shown in Fig. 8. When using 3×3 convolutional layer, the generated clean line drawings are frequently over simplified, so the details may be lost. On the contrary, using the 1×1 convolutional layer not only well simplifies the sketch, but also preserves the details.

Comparison on Feature Adoption. We further validate the effectiveness of multi-level perceptual loss. In Fig. 11b, measuring the perceptual loss on an optimal single layer fails to handle complicated sketchy strokes, and usually leads to oversimplification. For example, the head of the pirate in the first row and the right eye of the girl in the second row are now well simplified. In Fig. 11c, measuring the perceptual loss on the average of all layers cannot properly balance the importance of different layers of feature maps and leads to the simplification blurriness. The results are also quite unstable. For example, the weapon of the pirate is simplified into a line. In sharp comparison, our multi-level perceptual loss with automatically learned weights, shown as Fig. 11d, achieve the best visual quality in sketch simplification in terms of both simplification and detail preservation.

TABLE 1
Average Value Comparison of the Models Optimized with Responses on Different VGG Layers

Layer	Avg	Layer	Avg	Layer	Avg
ReLU1_1	2.6462	ReLU1_2	2.4762	-	-
ReLU2_1	2.7214	ReLU2_2	2.2281	-	-
ReLU3_1	2.9204	ReLU3_2	3.2638	ReLU3_3	3.3735
ReLU4_1	3.4219	ReLU4_2	3.7647	ReLU4_3	3.8095
ReLU5_1	**3.9137**	ReLU5_2	3.8296	ReLU5_3	3.6854

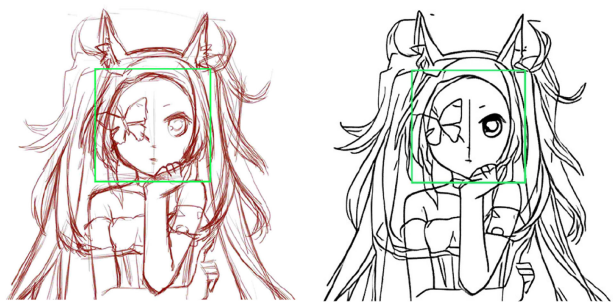
TABLE 2
Comparison with [5], [6] and the Model of [6] Retrained on Our Traning Data, Noted by [6]*

	Overall aesthetics	Content conformity	Tidiness	Average point
Simo-Serra et al. [5]	2.9974	3.2589	2.8769	3.0443
Simo-Serra et al. [6]	3.4012	3.5089	3.3974	3.4358
Simo-Serra et al. [6]*	3.0820	3.3269	3.0205	3.1431
SL	3.9000	3.9987	3.8346	3.9110
AL	4.0743	4.1230	4.0615	4.0862
AM (ours)	**4.2423**	**4.2115**	**4.2910**	**4.2482**

5.2.2 Comparison to Existing Methods

We compare our results with those of state-of-the-art methods including [5] and [6]. For fairness, we compare to [6] on both their fine-tuned one and the one retrained with our training data. Besides, we carry out this experiment in two ways. One is to compare the raster results directly derived from their networks, as shown in Fig. 12. The other is to compare the vectorized results in Fig. 13 since vectorization is applied as post-processing for [5], [6].

In Fig. 12, we can observe that for the extremely sketchy strokes with complicated structures (e.g., the face and cloth in first row, hands and book in second row) [5] is apt to output serious blurriness, and [6] leads to a mess of structures and hence loses its semantic meaning. Even if [6] retrained on our dataset, the above problems still exist. That is because these two existing methods simplify the sketches with less attention to the semantic understanding. However, our results can suppress the blurriness and well simplify extremely sketchy drawing faithfully to its semantic meaning in different scales, which is also verified in Fig. 13. But, it is shown in Fig. 13 that for [5] and [6], although vectorization



(a) Input (b) Ours

Fig. 9. Color-coded distribution of MOS scores for the second study. Mean value shown as red marker.

TABLE 3
Comparison on Time Statistics with [5] and [6] on the Implementation Details

	[5]	[6][a]	Our method
Input size	424 × 424	384 × 384	384 × 384
Batch size	6	16	6
Convergence (iterations)	600,000	150,000	50,000

a. Note that it uses a pretrained model of [5].

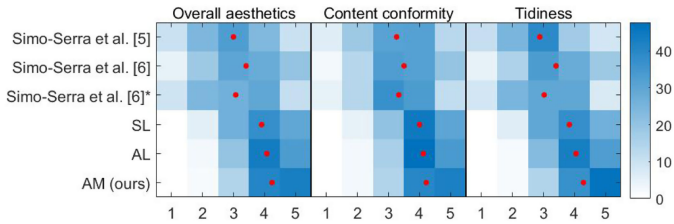


Fig. 10. Failure Case. There is an auxiliary line in the input (in green), and we cannot remove the auxiliary line as rough sketches.

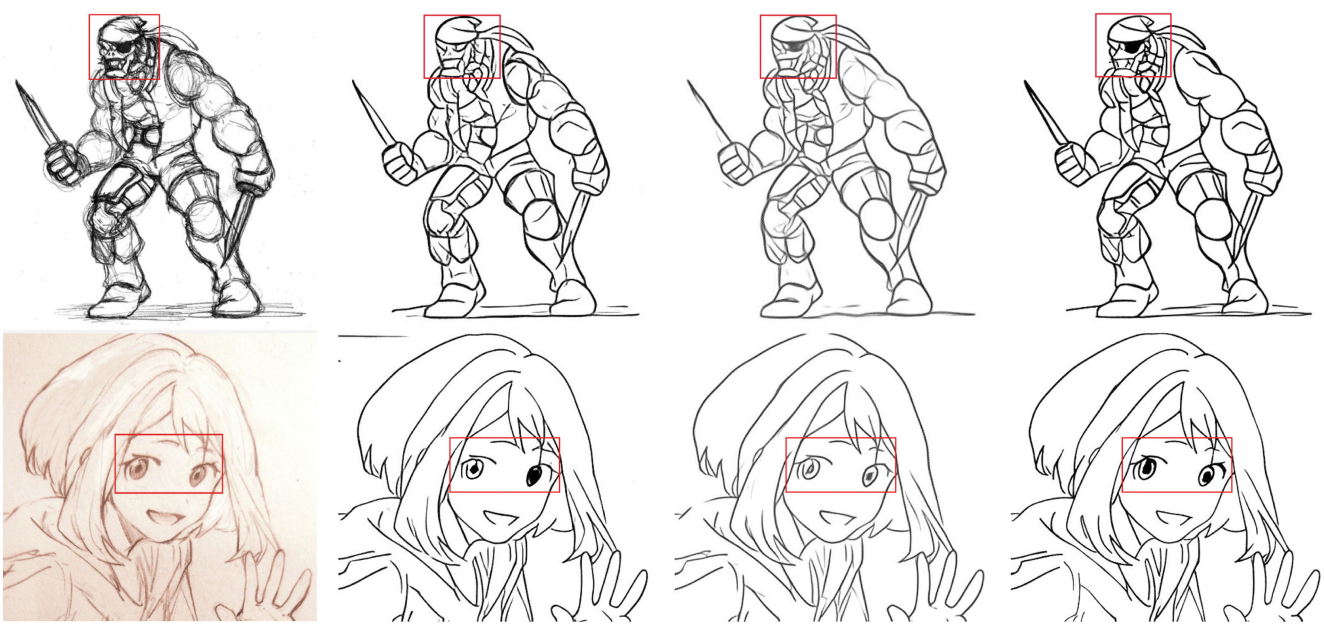
can remove the blurriness, it may drop some untraceable content in the direct raster results. More results are shown in Fig. 14. We can obviously see that under the uncontrolled lighting conditions, their methods fail to simplify the input and add additional artifacts, but ours can still obtain better results.

5.3 User Study

Due to the lack of effective quantitative measurement, we conduct a comprehensive user study to validate our work. We perform two perceptual user studies for a quantitative analysis of additional test data that is not part of our training set. Two user studies are conducted. The first determines the optimal VGG single layer for our ablation analysis. The second compares different configurations of our method to existing methods. We invited 30 participants (15 males and 15 females), aging from 20 to 30. All participants take part in both user studies.

VGG Layer Selection. First of all, we train 13 networks, each with a distinct single layer out of 13 fine-tune VGG layers. After that, we select 41 test sketches (21 of our proposed dataset and 20 collected from Simo-Serra et al. [6]) for the user study. With 41 inputs and 13 networks, we generate 533 output simplified sketches. A Mean Opinion Square (MOS) test is performed to determine the optimal VGG layer. All 533 simplified sketches are *randomly* presented to the participants one by one, together with the input sketch. The users are asked to rate each drawing in three aspects: overall aesthetics, tidiness of lines, and content conformity to the input. The range of rating is from 1 to 5, with 1 being the worst, and 5 the best. Table 1 shows the statistics using an average value for the three aspects, and detailed data is provided in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2019.2930512. The results of user study show that VGG layer (ReLU5_1) performs better than others from every aspect. Therefore we choose the ReLU5_1 VGG layer, as the baseline of our optimal single layer selection.

Effect of Different Layer Configurations. In this experiment, we feed the same 41 input sketches to three different configurations (*SL, AL, AM*) of our method and two state-of-the-art methods [5], [6]. Besides, we also compare the model of [6] retrained with our training data marked by [6]* to have a fair comparison. The statistics are shown in Table 2 and Fig. 9. Table 2 demonstrates the qualitative results of different configurations of our method and the state-of-the-art methods. We have the following observations. First, the scores of [5], [6] and [6]* are worse than that of *SL* which indicates that the perceptual loss is more valid to the task than pixel loss and discriminator loss. Second, comparing "*AM*" with "*SL*/*AL*", we can find that the network with automatically learned weights outperforms the network

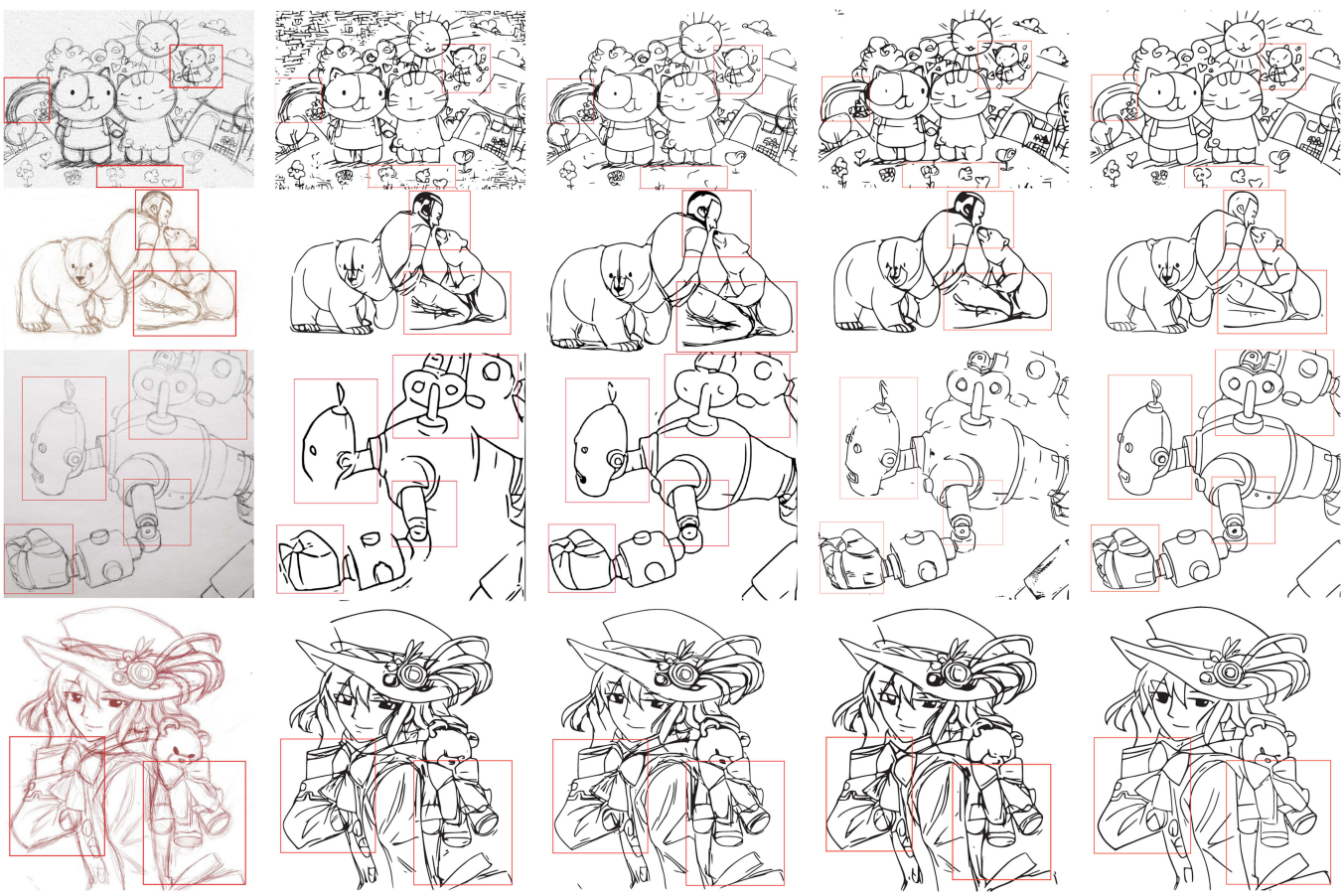


(a) Input (b) SL (c) AL (d) Ours (AM)

Fig. 11. Ablation study on the improvement from adoption of multi-level perceptual loss and feature adjustment.

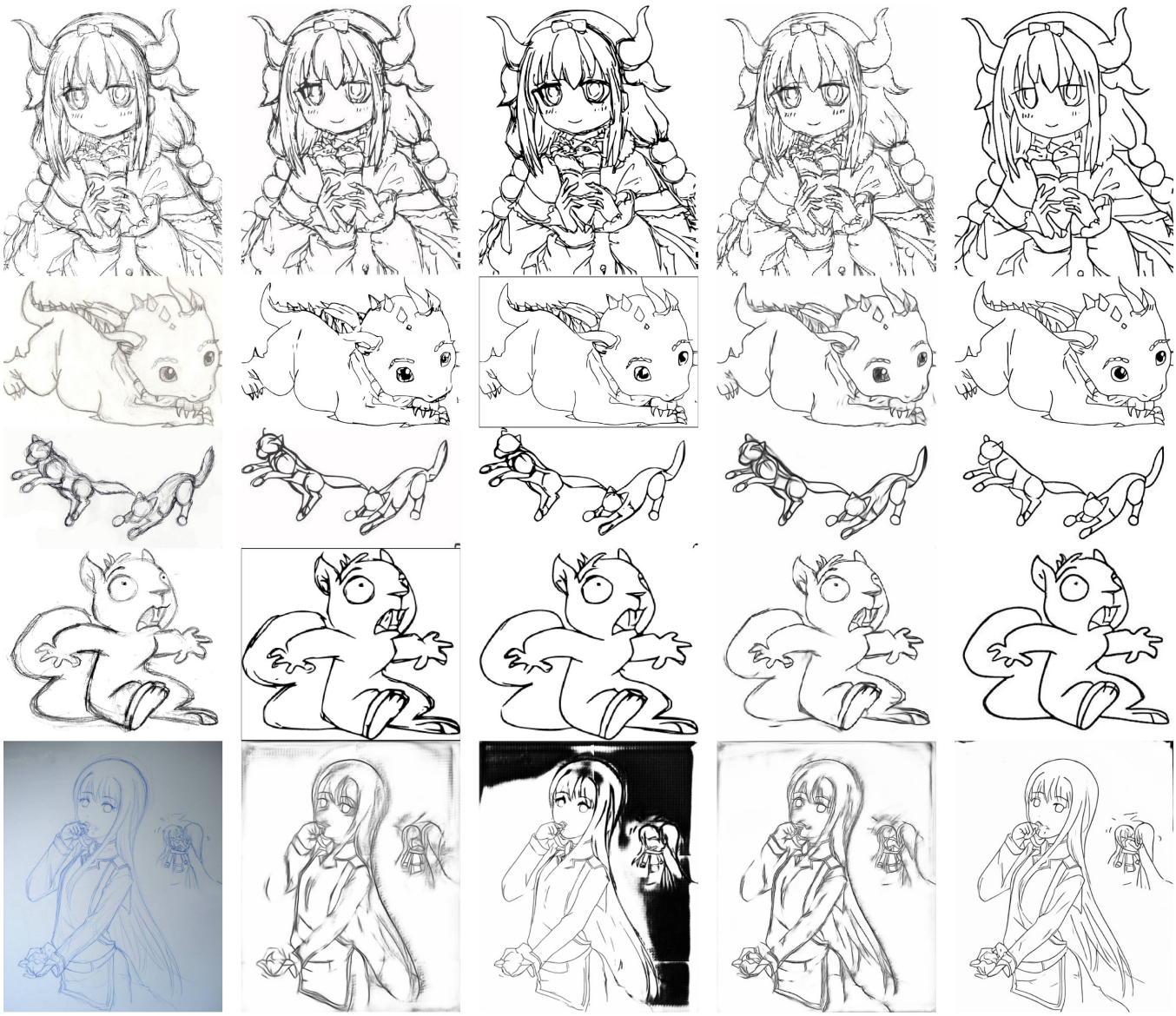
(a) Input (b) [5]* (c) [6] (d) [6]* (retrained) (e) Ours

Fig. 12. Further comparison of the original results without vectorization between ours and [5], [6]. Please note that the original method of [5] include vectorization as part of post-processing and [6]* represents the model is retrained on our training data.



(a) Input (b) [5] (c) [6] (d) [6]* (retrained) (e) Ours

Fig. 13. Further comparison of the vectorized results between ours and [5], [6], please note that [6]* represents the model is retrained on our training data.



(a) Input (b) [5] (c) [6] (d) [6]* (retrained) (e) Ours

Fig. 14. More comparisons of the non-vectorized results between ours and [5], [6], please note that [6]* represents the model is retrained on our training data. Among them, the input image of the last line is taken with a camera.

without it, which demonstrates the effectiveness of our proposed multi-layer perceptual network.

We further compare the color-coded distribution of MOS scores between our model and [6] in terms of each aspect, with dependent T-test and obtain p-values which are much smaller than 0.05 respectively, indicating that results are significantly different.

5.4 Time Statistics

We conduct our experiments on a PC with Intel i7 CPU, 32G RAM equipped with single 1080Ti GPU. First, we use the method of GoogleNet [28] to fine-tune the pre-trained model of VGG16 with batch normalization [7] on our training data. Next, we freeze the fine-tuned VGG16 and optimize the multi-layer discriminator and simplification network simultaneously in an end-to-end manner using the Adam solver with a learning rate of 0.001 and batch size of 6. It takes 22 hours for the training process. The training process takes about 50,000 iterations to converge, which is faster than the existing methods (600,000 iterations in [5] and 150,000 iterations in [6]). A detailed comparison is given in Table 3.

5.5 Limitations

Our method has the following limitations. First, due to different sketching style, some artists may introduce auxiliary lines, shown in Fig. 10a. The auxiliary lines are not included during our training. Hence our network cannot remove those auxiliary lines as rough sketches represented as Fig. 10b. Second, it remains difficult when the resolution of the image is too low; the network inevitably oversimplifies the sketch and loses the details.

6 Conclusion

In this paper, we present a simple CNN-based method for achieving high-quality sketch simplification. Our main

contribution lies in the introduction of multi-layer discriminator with an automatic adjustment mechanism for balancing different feature layers. Since we employ all-layer features from the fine-tuned VGG which are reasonably and automatically fused with the help of multi-layer discriminator, our method can capture semantically important structure in both local and global scale. Hence, it solves the problems of blurriness and performs better on preserving the details and the semantic structure. We further proposed a new dataset augmentation approach to handle the uncontrolled lighting conditions that may be introduced during casual photography. Convincing results are obtained in all tested cases.

We may further explore if other extraction models can bring more improvements. We do not have a sophisticated binarization scheme when vectorization is needed. Further study on this binarization is needed.

Acknowledgments

The work is supported by NSFC (Grant No. 61772206, U1611461, 61472145), Guangdong R&D key project of China (Grant No. 2018B010107003), Guangdong High-level personnel program (Grant No. 2016TQ03X319), Guangdong NSF (Grant No. 2017A030311027), and Guangzhou key project in industrial technology (Grant No. 201802010027).

References

[1] B. Wilson and K.-L. Ma, "Rendering complexity in computer-generated pen-and-ink illustrations," in *Proc. 3rd Int. Symp. Non-Photorealistic Animation Rendering*, 2004, pp. 129–137.

[2] Y. Qi, Y. Z. Song, T. Xiang, H. Zhang, T. Hospedales, Y. Li, and J. Guo, "Making better use of edges via perceptual grouping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1856–1865.

[3] X. Liu, T.-T. Wong, and P.-A. Heng, "Closure-aware sketch simplification," *ACM Trans. Graph.*, vol. 34, no. 6, 2015, Art. no. 168.

[4] J.-D. Favreau, F. Lafarge, and A. Bousseau, "Fidelity vs. simplicity: A global approach to line drawing vectorization," *ACM Trans. Graph.*, vol. 35, no. 4, 2016, Art. no. 120.

[5] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa, "Learning to simplify: Fully convolutional networks for rough sketch cleanup," *ACM Trans. Graph.*, vol. 35, no. 4, 2016, Art. no. 121.

[6] E. Simo-Serra, S. Iizuka, and H. Ishikawa, "Mastering sketching: Adversarial augmentation for structured prediction," *ACM Trans. Graph.*, vol. 37, no. 1, 2018, Art. no. 11.

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

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, May 7–9, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[9] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4681–4690.

[10] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6882–6890.

[11] Q. Yang, P. Yan, M. K. Kalra, and G. Wang, "CT image denoising with perceptive deep neural networks," *CoRR*, vol. abs/1702.07019, 2017. [Online]. Available: http://arxiv.org/abs/1702.07019

[12] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6836–6845.

[13] P. Zhang, D. Wang, H. Lu, H. Wang, and R. Xiang, "Amulet: Aggregating multi-level convolutional features for salient object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 202–211.

[14] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018. doi: 10.1109/TPAMI.2017.2699184

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[16] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 447–456.

[17] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 577–593.

[18] S. H. Bae, R. Balakrishnan, and K. Singh, "ILoveSketch: As-natural-as-possible sketching system for creating 3D curve models," in *Proc. ACM Symp. User Interface Softw. Technol.*, 2008, pp. 151–160.

[19] J. Arvo and K. Novins, "Fluid sketches: Continuous recognition and morphing of simple hand-drawn shapes," in *Proc. ACM Symp. User Interface Softw. Technol.*, 2000, pp. 73–80.

[20] B. Preim and T. Strothotte, "Tuning rendered line-drawings," in *Proc. Winter School Comput. Graph.*, 1995, pp. 228–238.

[21] S. Grabli, F. Durand, and F. X. Sillion, "Density measure for line-drawing simplification," in *Proc. 12th Pacific Conf. Comput. Graph. Appl.*, 2004, pp. 309–318.

[22] A. Shesh and B. Chen, "Efficient and dynamic simplification of line drawings," *Comput. Graph. Forum*, vol. 27, pp. 537–545, 2008.

[23] X. Hilaire and K. Tombre, "Robust and accurate vectorization of line drawings," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 890–904, Jun. 2006.

[24] G. Noris, A. Hornung, R. W. Sumner, M. Simmons, and M. Gross, "Topology-driven vectorization of clean line drawings," *ACM Trans. Graph.*, vol. 32, no. 1, 2013, Art. no. 4.

[25] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4076–4084.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.

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

[28] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, "The sketchy database: Learning to retrieve badly drawn bunnies," *ACM Trans. Graph.*, vol. 35, no. 4, 2016, Art. no. 119.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[30] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 649–666.

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

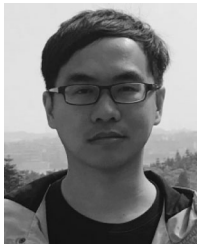
Xuemiao Xu received the BS and MS degrees in computer science and engineering from the South China University of Technology, in 2002 and 2005 respectively, and the PhD degree in computer science and engineering from the Chinese University of Hong Kong, in 2009. She is currently a professor with the School of Computer Science and Engineering, South China University of Technology. Her research interests include object detection, tracking, recognition, and image, video understanding and synthesis, particularly their applications in the intelligent transportation. She is a member of the ACM.

Minshan Xie received a bachelor's degree in software engineering and a master's degree in computer science and technology from South China University of Technology in 2015 and 2018, respectively. She is currently working toward the PhD degree in the Department of Computer Science and Engineering, the Chinese University of Hong Kong. Her research interests include computer graphics, image processing, computer vision and deep learning.

Peiqi Miao received the bachelor of science in management accounting degree in the business administration from the South China University of Technology, China, in 2016. She is currently working toward the master's degree in the School of Computer Science and Engineering, South China University of Technology. Her research interests include the application and research of deep learning in anime.

Wei Qu received the BEng and BA degrees from the Dalian Jiao Tong University, China, in 2018. She is currently working toward the master's degree in the School of Computer Science and Engineering, South China University of Technology. Her research interests include deep learning and computer graphics and computer vision.

Wenpeng Xiao is working toward the graduate degree in the Computer Science and Technology from the South China University of Technology. His research interest includes computer vision, animations and deep learning.



Huaidong Zhang received the BEng degree in the Computer Science and Technology from the South China University of Technology, China, in 2015. He is currently working toward the PhD degree in the School of Computer Science and Engineering, South China University of Technology. His research interests include computer vision, image processing, computer graphics and deep learning.

Xueting Liu received the BEng degree from Tsinghua University and the PhD degree from the Chinese University of Hong Kong, in 2009 and 2014 respectively. She is currently an assistant professor with the School of Computing and Information Sciences, Caritas Institute of Higher Education. Her research interests include computer graphics, computer vision, machine learning, computational manga and anime, and non-photorealistic rendering.

Tien-Tsin Wong received the BSc degree in computer science from the Chinese University of Hong Kong, in 1992, and the MPhil and PhD degrees in computer science from the same university, in 1994 and 1998 respectively. In August 1999, he joined the Computer Science & Engineering Department, Chinese University of Hong Kong. He is currently a professor. He is a core member of Virtual Reality, Visualization and Imaging Research Centre in the Chinese University of Hong Kong. His main research interests include computer graphics, computational manga, precomputed lighting, image-based rendering, GPU techniques, medical visualization, multimedia compression, and computer vision. He is a senior member of the ACM.