浙江大學
ZHEJIANG UNIVERSITY

Reproduction of paper <Rough sketch simplification –SimoSerra et al.>

小组成员：王海纳（答辩人）、卜一轩

CONTENTS

00 Overview

01 Paper Reviews

02 Reproductions

03 Analysis

04 Discussion

00

Overview

Overview of our presentation

00 Overview (1/3)

We've reproduced the paper "Rough sketch simplification" by Edgar Simo-Serra Et al, published in SIGGRAPH 2016.

Our work includes:

- Rewrite the author's .lua code into python, and reproduce every figures in the paper with our own data, comparing them with original figures and make analysis & discussions.
- Design some tests to validate the main contribution of the paper.
- Make further discussions about paper from today's aspect.

00 Overview (2/3)

务求实学，存是去非

Source

Ours-after

正其谊，不谋其利；明其道，不计其功

00 Overview (3/3)

浙江大学 ZHEJIANG UNIVERSITY

References to author's work

Contents	Author	Our action
Training Code	Provided in Lua.torch on Github Other's provided modified pytorch code.	Rewrite it in Pytorch(torch==2.0.1, torchvision==0.15.2)
Dataset & Data construction code	Not provided	Prepared from various sources and construct data by ourselves
Further treatments' Code	Not provided	explored by ourselves

务求实学，存是去非

正其谊，不谋其利，明其道，不计其功

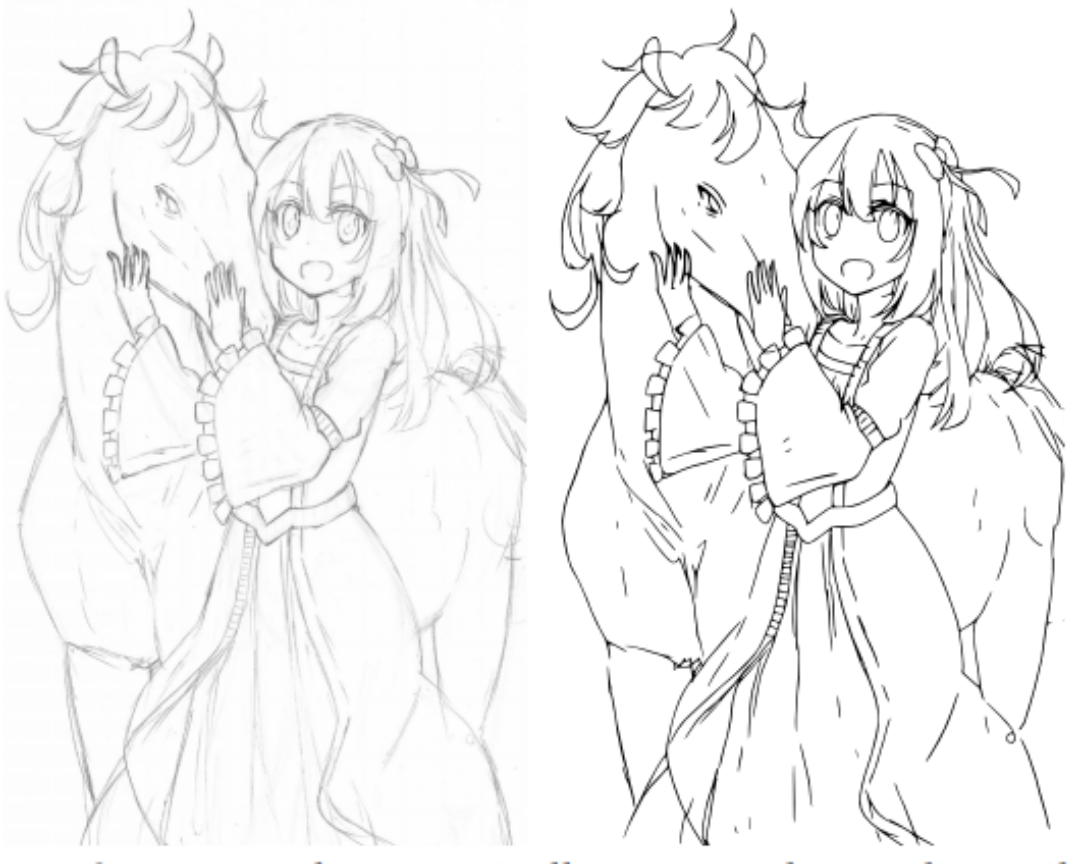
01

Paper review

Review of the original paper.

(All pictures in this sections are from the original paper if non mentioned)

01 Paper Reviews(1/8)



The overall contribution of this paper:

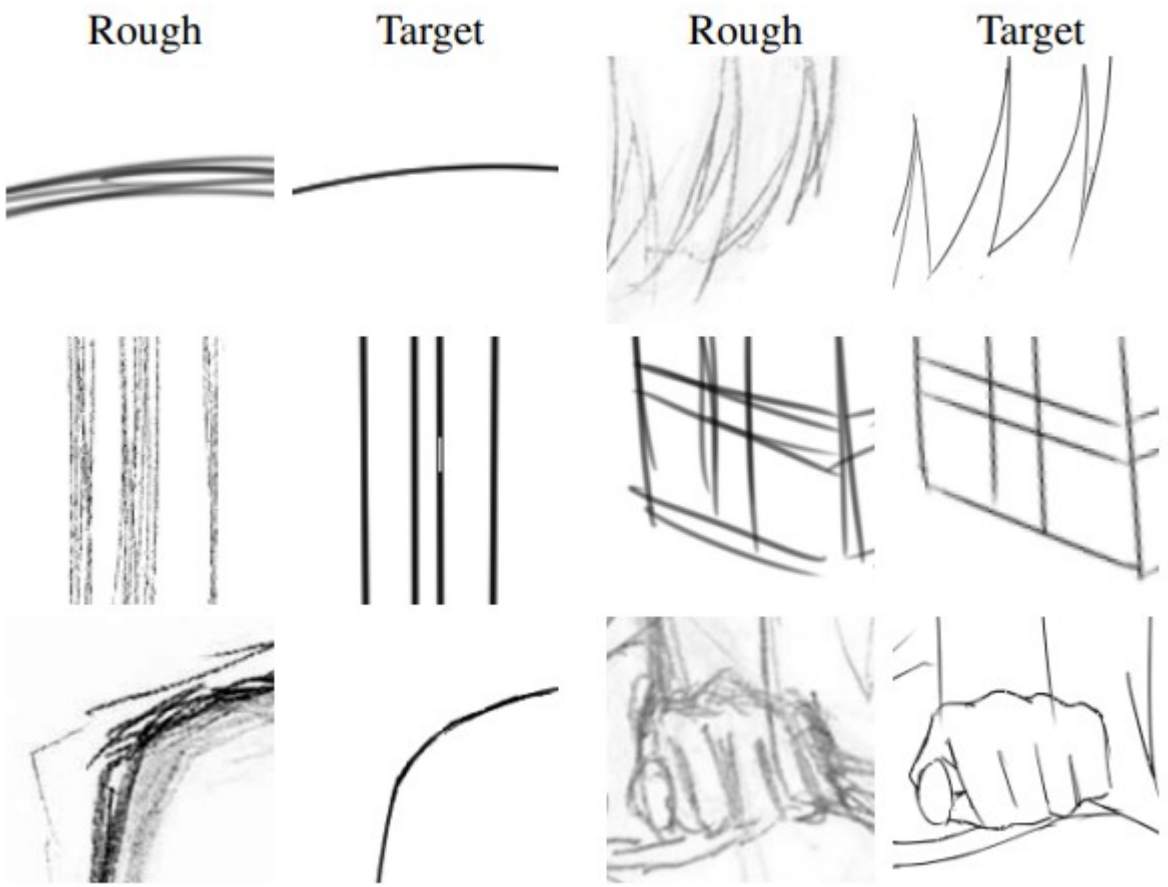
1)The first sketch simplification approach that is optimized to directly operate on raster images of rough sketches.

2)A novel fully Convolutional Neural Network architecture that can simplify sketches directly from images of any resolution.

3)An efficient approach to learn the sketch simplification network.

4)A dataset for large-scale learning of sketch simplification. (Not provided)

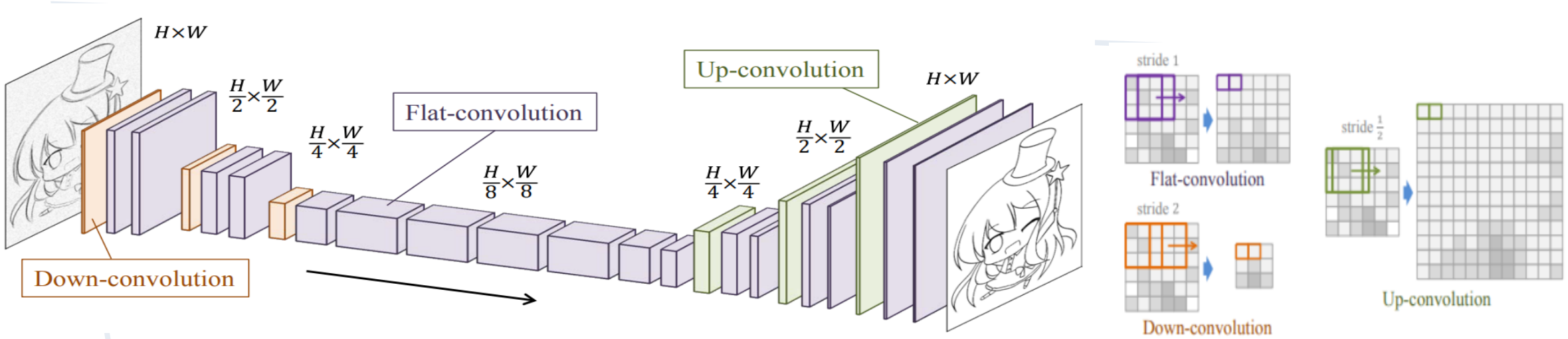
01 Paper Reviews(2/8)



Examples of anime sketch simplifications.

- It's quite common for **multiple lines to have to be collapsed into a single line**.
- The **intensity of different input lines vary greatly** even within the same image.

01 Paper Reviews(3/8)



Overview of the convolutional network structure. (U-net like)

The novel points(at 2016) the author argues:

- Accumulated 8*8 pixel pooling in form of down-convolutions, which enables larger range of output image resolution.

01 Paper Reviews(4/8)

One of the main recent innovations that have allowed the training of such deep models as the one we present from scratch are batch normalization layers [Ioffe and Szegedy 2015]. They consist of simply keeping a running mean and standard deviation of the input data, and using them to normalize the input data. The output of these layers roughly has a mean of 0 and a standard deviation of 1. The running mean is only updated during training and is kept fixed during evaluation. Batch normalization layers also have two additional learnable parameters that serve to scale the normalized output and add a bias:

$$y_{BN}(x) = \frac{x - \mu}{\sqrt{s^2 + \epsilon}}\gamma + \eta \,, \tag{7}$$

where μ is the running mean, s is the running standard deviation, ϵ is a constant for numerical stability, and γ and η are learnable parameters. We use these layers after all convolutional layers except

For learning the weights of the models, we rely on the ADADELTA algorithm [Zeiler 2012]. The main advantage of this approach is that it does not require explicitly setting a learning rate, which is a non-trivial task. ADADELTA has been shown to generally converge to similar solutions as other algorithms, however, it will take a longer time to converge in comparison with optimally tuned learning rates. For training a sketch model, we tried various other optimizers and found that the result did not change significantly, while other optimizers have the added complexity of choosing a learning rate scheduler. ADADELTA consists of keeping a running mean of

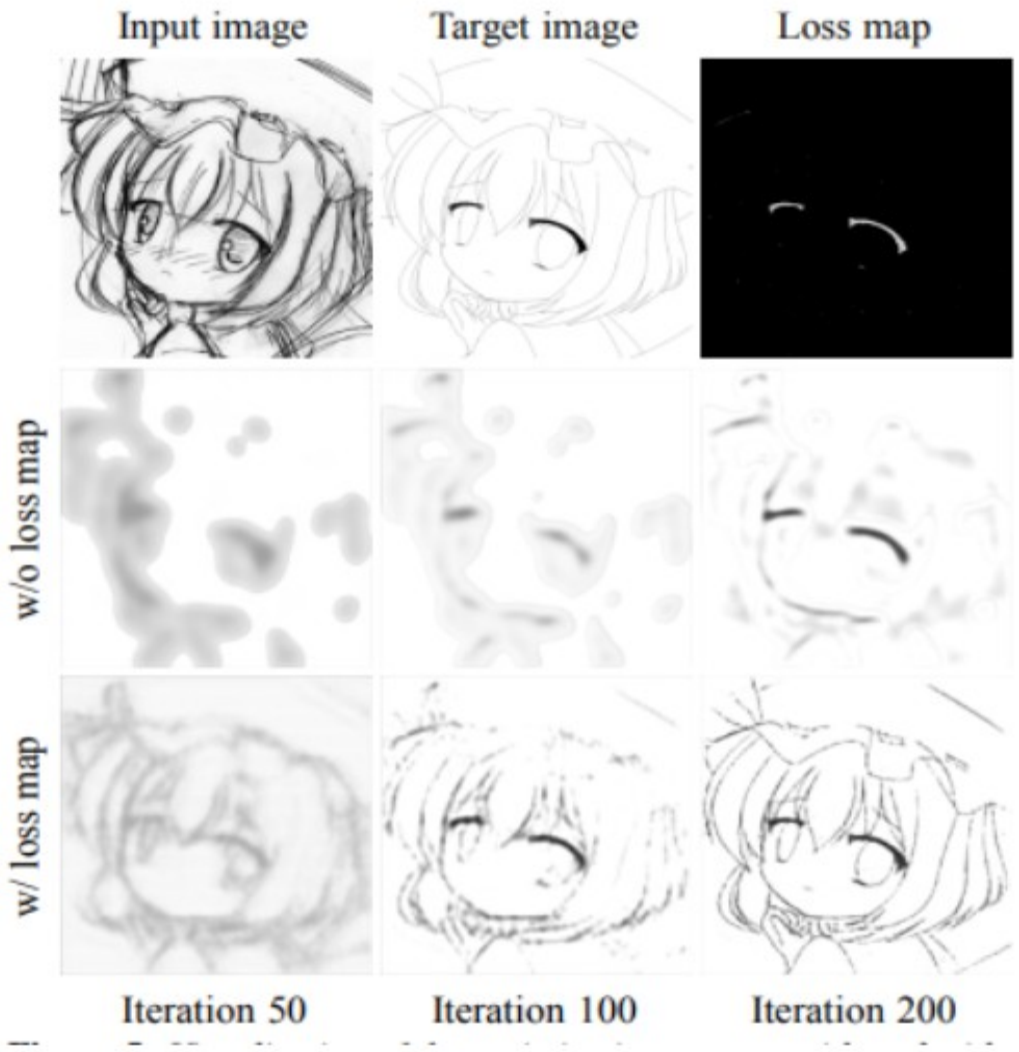
the square of the gradients and the square of the updates, which are used to determine the learning rate. An update of the parameters of the model θ then becomes,

$$\theta_{t+1} = \theta_t + \Delta\theta_t = \theta_t - \frac{\mathrm{RMS}[\Delta\theta]_{t-1}}{\mathrm{RMS}[\delta\theta]_t}\delta\theta_t \,, \tag{8}$$

Besides convolutions, the author stresses batch normalization and uses ADADELTA as the learning method.

However these are just common methods in pytorch now…

01 Paper Reviews(5/8)



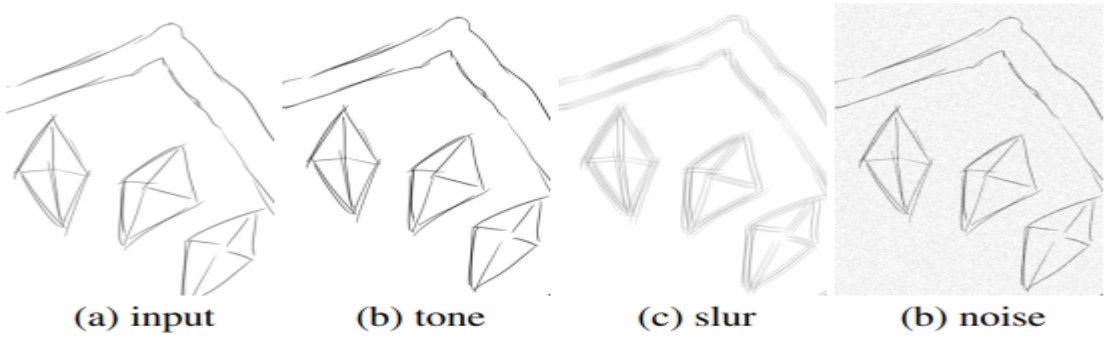
The author use the folloing loss map to calculate the loss function.

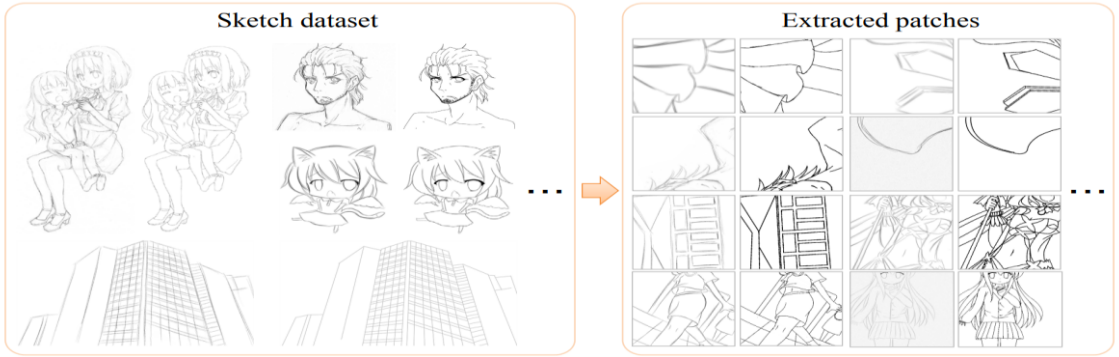
$$l(Y, Y^*, M) = \| M \odot (Y - Y^*) \|_{\mathrm{FRO}}^2 ,$$

$$M(u,v) = \begin{cases} 1 & \text{if } I(u,v) = 1 \\ \min\left(\alpha \exp\left(-H(I,u,v)\right) + \beta, 1\right) & \text{else} \end{cases}$$

Y is current image, Y* is ground truth, I(u,v) denotes a pixel on Y.
H(I,u,v) is the "*value of the bin of the local normalized histogram constructed using all pixels within Dh pixels from center using Bh bins in which I(u,v) falls into*". This will be discussed Further.

01 Paper Reviews(6/8)





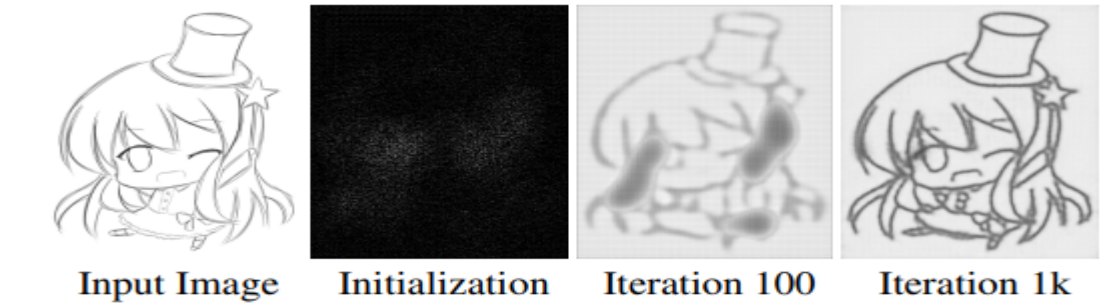
The author's team use a special method to design and use the dataset:

Inverse dataset generation

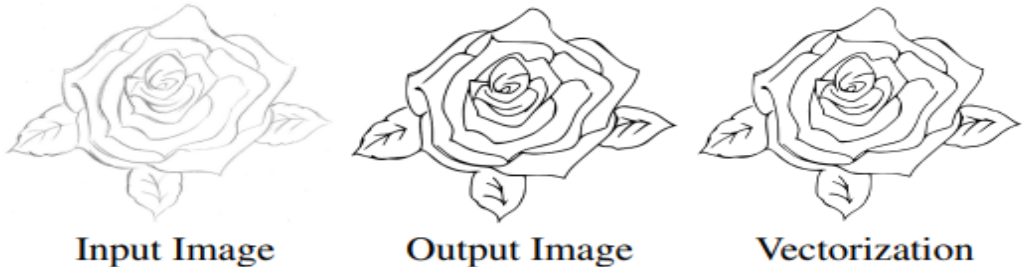
Instead of asking artists to draw rough sketch and clean it up, the author only ask for clean sketch and use tools like Adobe Photoshop to add noises manually.

After this, they randomly pick a fixed-size patches from every picture to use for training.

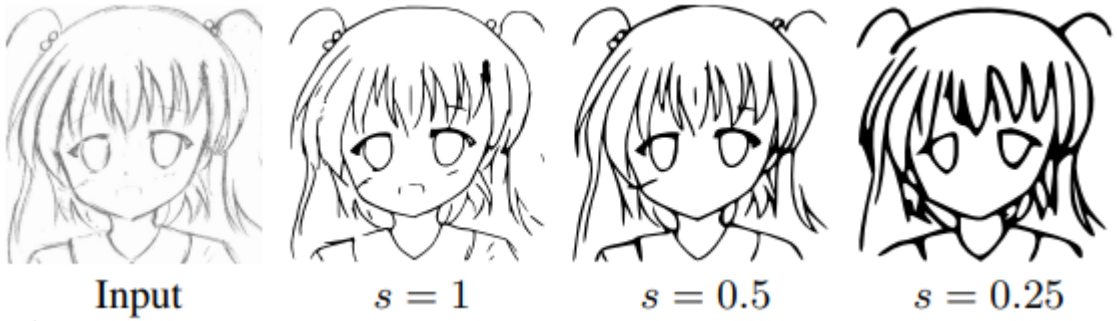
01 Paper Reviews(7/8)



The training results with different number of iterations.



The author further attempted Vectorization using *"automatic publically available tools with default parameters"*



The author also argue that the extend of simplification can be controlled by scaling the pictures.

01 Paper Reviews(8/8)

	Ours	Live Trace	Potrace
Score	**4.53**	2.94	2.80
vs Ours	-	2.5%	2.8%
vs Live Trace	**97.5%**	-	30.3%
vs Potrace	**97.2%**	69.7%	-

ually set the threshold for each image to obtain best results. Comparison was done in two formats: (a) comparing two processed images to see which is better, and (b) ranking a processed image on a scale of 1 to 5.

We used 19 users for both cases, 10 of whom had significant experience in sketch drawing. Results are shown in Table 2. We can see that, when compared to the other approaches, our model was considered better in over 97% of the cases. Furthermore, in absolute terms, our approach was ranked 4.53 on a scale of 1 to 5. We found no significant differences between the naïve and expert users.

Obviously objective scores cannot evaluate the result well.

So the author designed questionnaires whose question includes “rate from 1-5” and “which one is better among the 3”.

The questionnaires are delivered to 19 users, among which 10 are experts.

The score of their method seemed quite competitive, and did not vary between experts and amateurs.

02

Reproduction

The work we have done during reproductions.

02-1 Reproduction overview(1/1)

Some parts of the method is novel at 2016 but common nowadays, such as the BatchNorm layer and adam optimizer.

So our rewritten code is kind of simpler than the author's lua code and finish rerunning every pictures appeared in the paper.

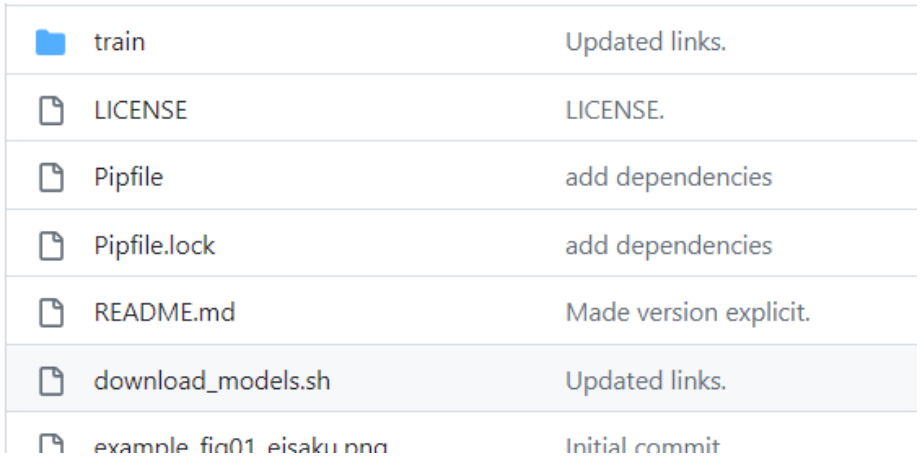
But we've still find some challenges after going through the paper, mainly if the highlights in the paper make sense:

1) In the learning stage: What's the power of the loss function?

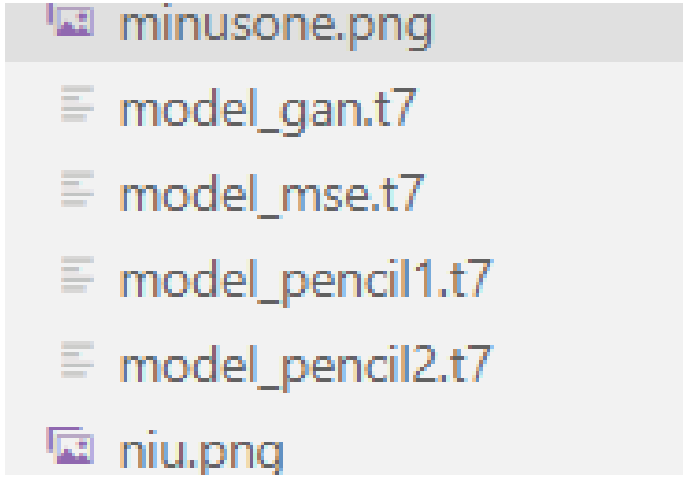
2) In the dataset: What's the power of the Inverse Dataset generation?

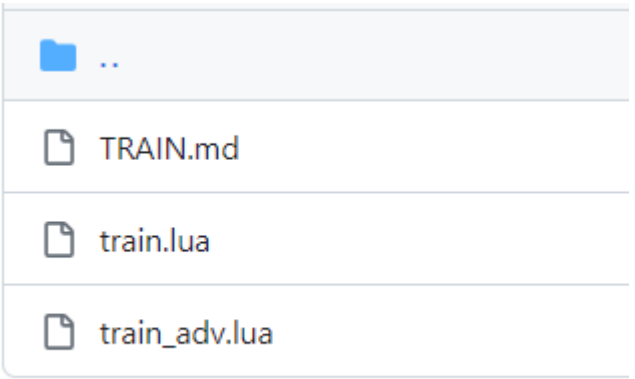
So we've designed further comparison experiments to test these.

02-2 Main Reproduction results(1/10)



train	Updated links.
LICENSE	LICENSE.
Pipfile	add dependencies
Pipfile.lock	add dependencies
README.md	Made version explicit.
download_models.sh	Updated links.
example_fig01_eisaku.png	Initial commit

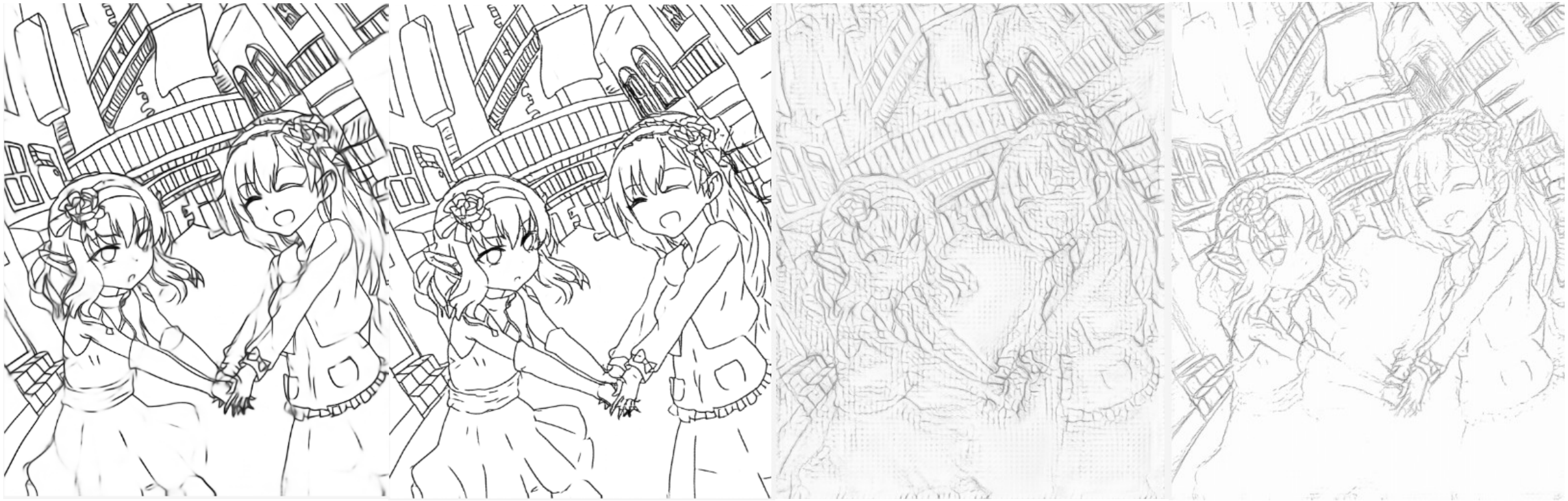




The author provided multiple models without full descriptions: there are “gan”,”pencil1/2” model but the original paper did not mention them at all. And in these new models, “model_gan.t7” perform quite better.

So we only attempt to reproduct the original model “model_mse.t7”, and use others to generate groundtruth.

02-1 Main Reproduction results(2/10)



From left to right: mse, gan, pencil1, pencil2

02-2 Main Reproduction results(3/10)

Author's picture A

Source

Paper

Ours

02-2 Main Reproduction results(4/10)

Author's picture B

Source

Paper

Ours

02-2 Main Reproduction results(5/10)

Author's picture C

Source

Paper

Ours

02-2 Main Reproduction results(6/10)

Author's picture D

Source

Paper

Ours

02-2 Main Reproduction results(7/10)

Our picture A

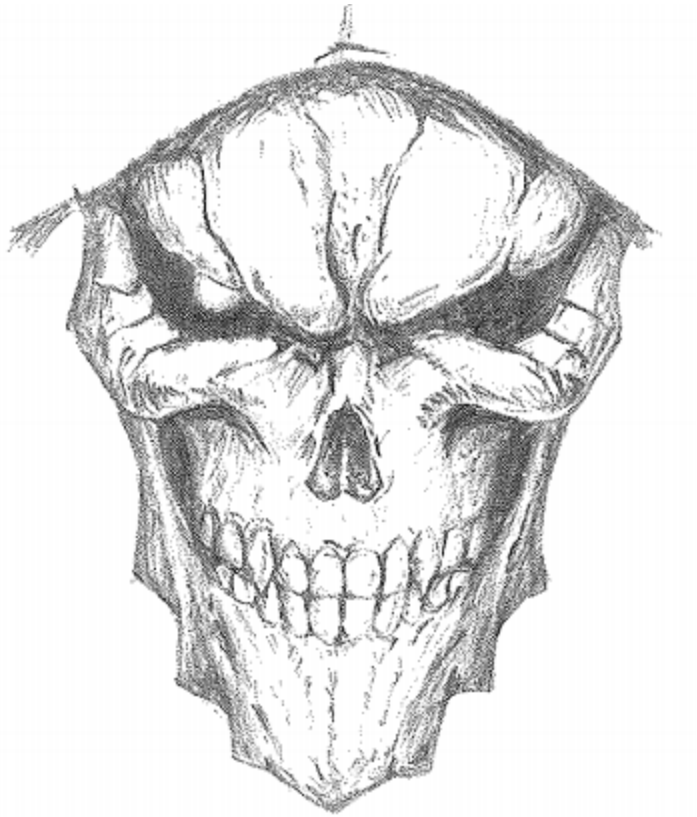
Source

Paper

Ours

02-2 Main Reproduction results(8/10)

Our picture B



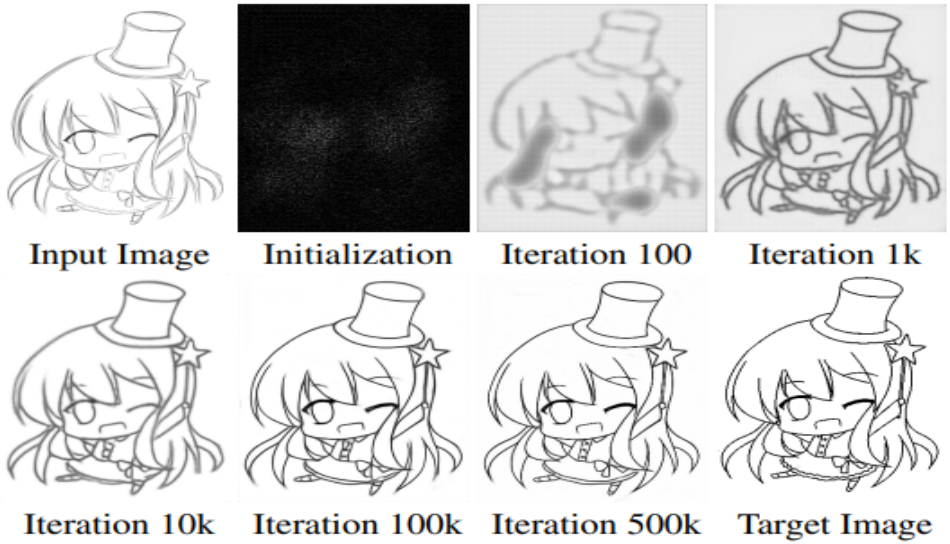
Source

Paper

Ours

02-2 Main Reproduction results (9/10)

Change during iterations:



Source

Ours

02-2 Main Reproduction results (10/10)

The effect after vectorizations:

Source

Ours-before

Ours-after

02-3 About the loss function(1/3)

The loss function author use is shown as follows:

$$l(Y, Y^*, M) = ||M \odot (Y - Y^*)||_{FRO}^2$$

$$M_{(u,v)} = \begin{cases} 1, & I(u,v) = 1 \\ min(\alpha exp(-H(I,u,v)) + \beta, 1), & otherwise \end{cases}$$

Where H(I,u,v) is the value of the bin of the local normalized histogram in which the pixel I(u, v) falls into. The histogram is constructed using all pixels within Dh pixels from the center using Bh bins. The author set Dh=2, Bh=10.
By the code, we've learnt that H(I,u,v) actually equals to

$$\frac{\#(pixels\ same\ to\ I(u,v)\ with\ the\ first\ demical)}{\#(pixels\ in\ the\ neighborhood)}$$

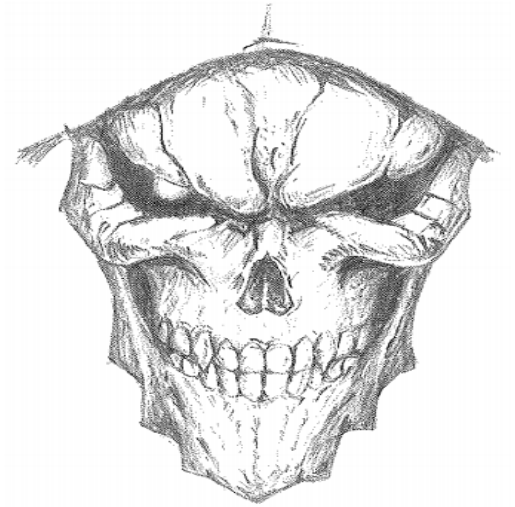
02-3 About the loss function(2/3)

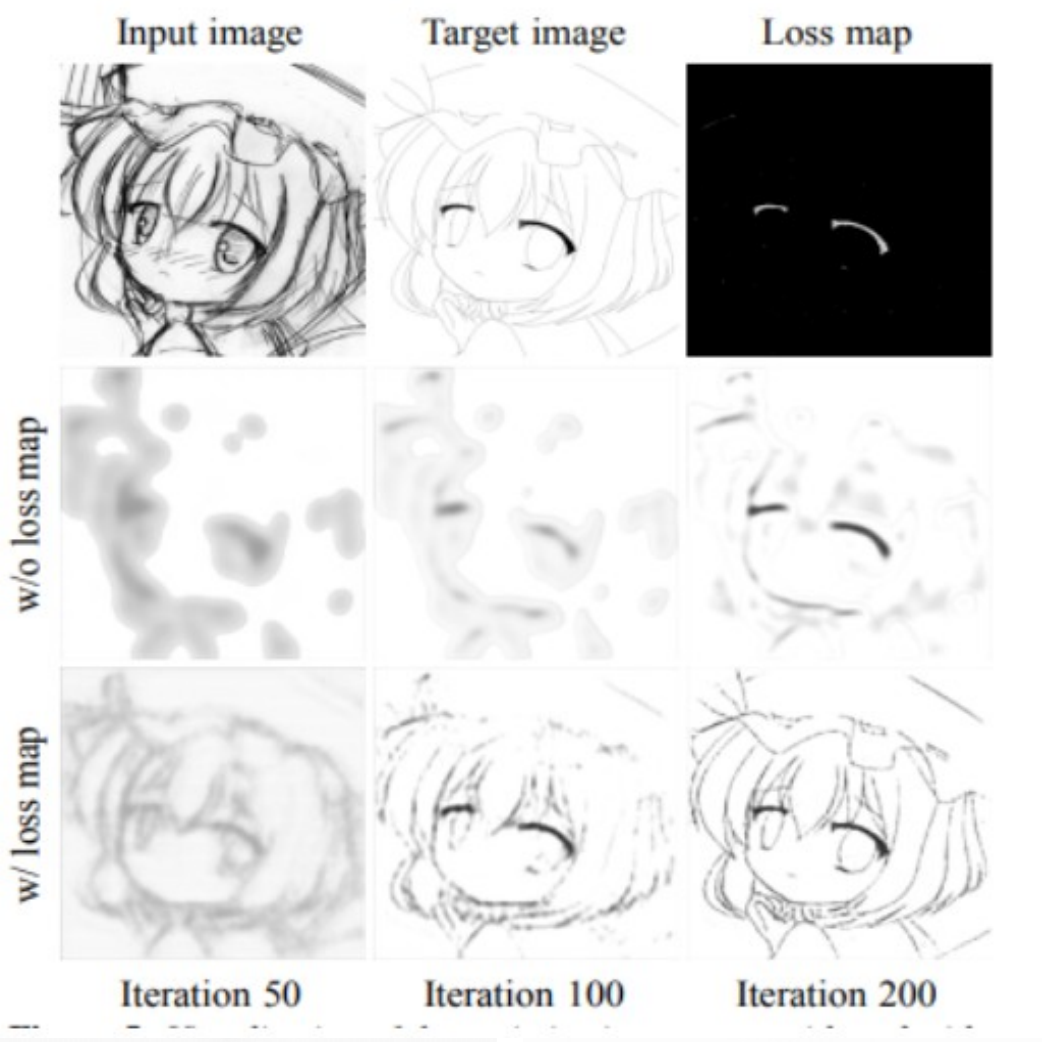
$$\frac{\#(pixels\ same\ to\ I(u,v)\ with\ the\ first\ demical)}{\#(pixels\ in\ the\ neighborhood)}$$

For loops is too slow in python, but vectorize this formula is skillful.
Our solution is to calculate (2Dh+1)(2Dh+1) maps, all generated by a shift within the neighbor field, and calculate their contribution one by one with np.select().

We have tested the effect of this loss function by comparison.

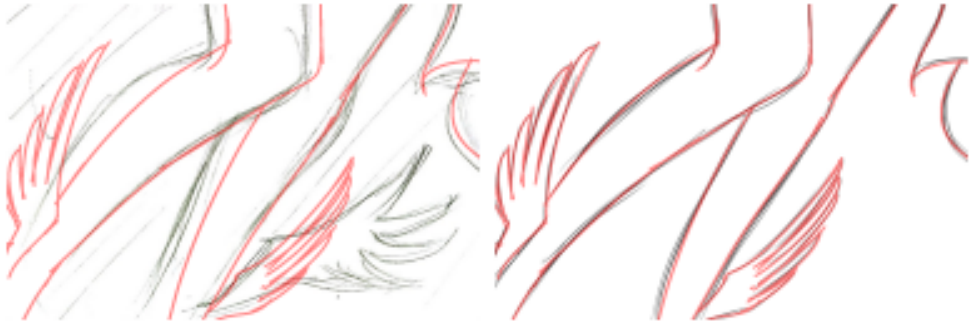
02-3 About the loss function(3/3)



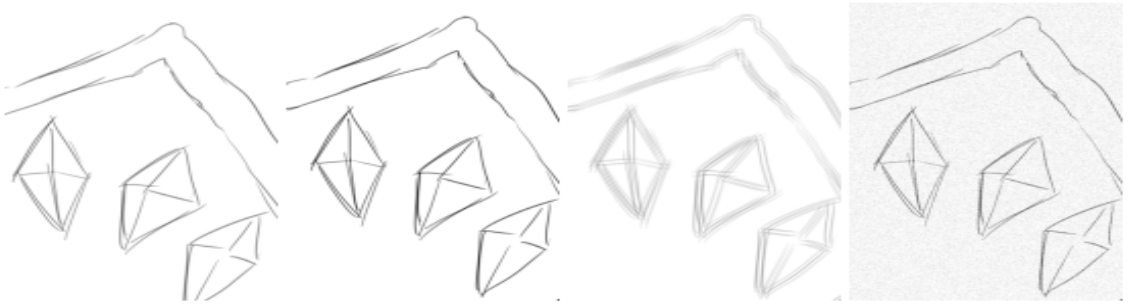


Training with this single image by 50, 100, 200epoch from left to right.

02-4 Dataset generation (1/9)



(a) Direct (b) Inverse



(a) input (b) tone (c) slur (b) noise

Several peculiarities the inverse dataset bears:

1) Small
2) Manual noise.
3) Low noise.
4) Random patches.

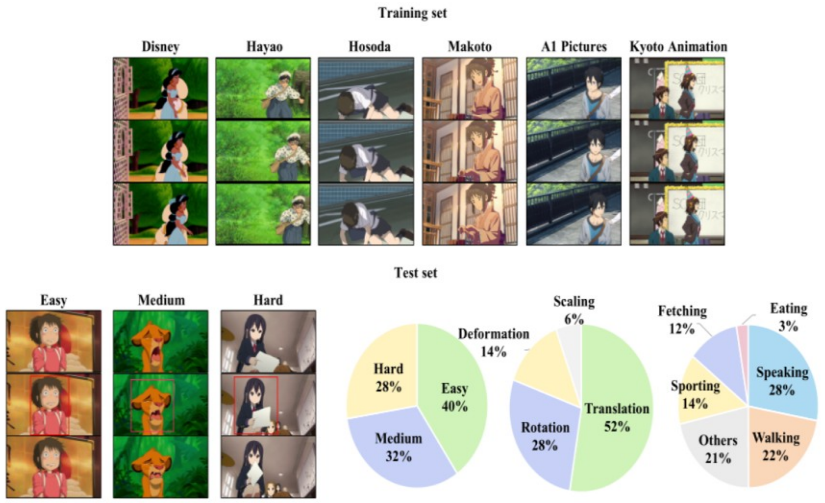
For 2, 3, 4, which are sensitive?

02-4 Dataset generation (2/9)

We've designed tests to validate the sensitivity of the dataset's features.

The sources of our dataset:

1) ATD-12K. (left). Containing 12000 * 3 cartoon images from Disney & Kyoto anime.[Siyao et al, **Deep Animation Video Interpolation in the Wild**, CVPR2021]
2) Layout images from **sakugabooru.com** or **baidu.com** (mid)
3) Hand-drawn image by ourselves(right).



02-4 Dataset generation (3/9)

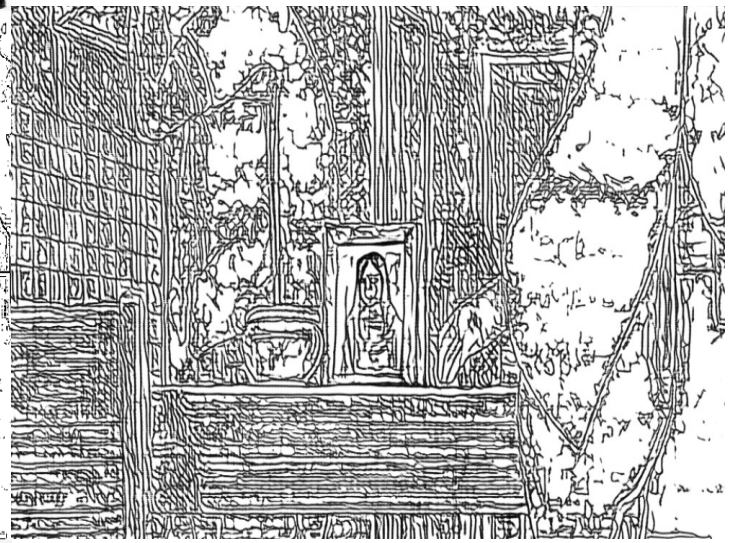
Since the author's (mysterious) model is provided and has spectacular performance, it may provide layouts and our pencil-drawn image with the help of some extra binarizations.

For Example: left->author model -> global binarization with threshold=70->right

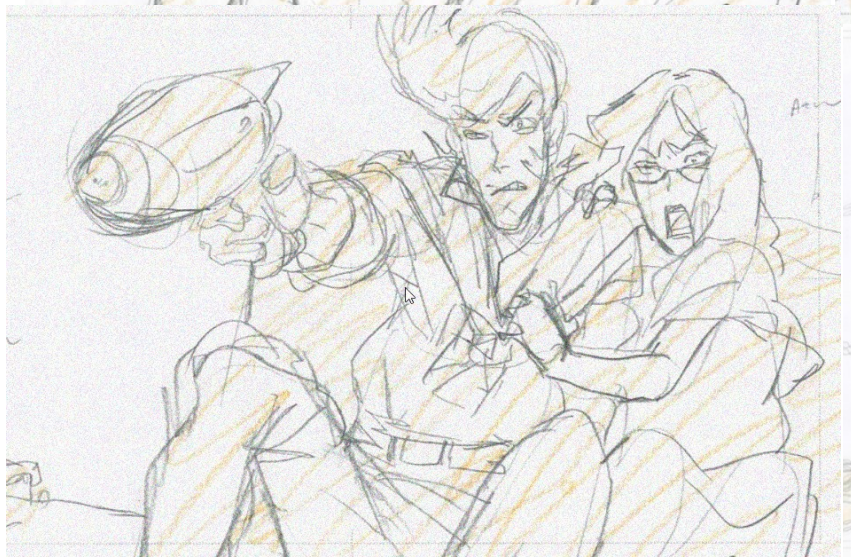
02-4 Dataset generation (4/9)

However the model is not designed to deal with colorful images.

Luckily, local binarization makes sense.



02-4 Dataset generation (5/9)



For noises, random noise(left-down) and tones(right-up) can be added by program automatically.

However slur (right-down)is hard to control, so we added it by PS.

02-4 Dataset generation (6/9)

Case ID	Original Image	Ground Truth
1	800 image from ATD-12K after local-binarization	The result after passing local-binarization, author's model and global binarization in a row
2	73*4 image: 73 layout images and their 73*3 copies with 3 kinds of noises respectively	The result after passing the author's model and global binarization in a row
3	73*2 image: original & slur	Same as above
4	73*2 image: original & random noise	Same as above
5	Same as 2, but without random patches	Same as 2

1 vs 2: The importance of manual noise

3 vs 4: The importance of limited noise

2 vs 5: The importance of patches

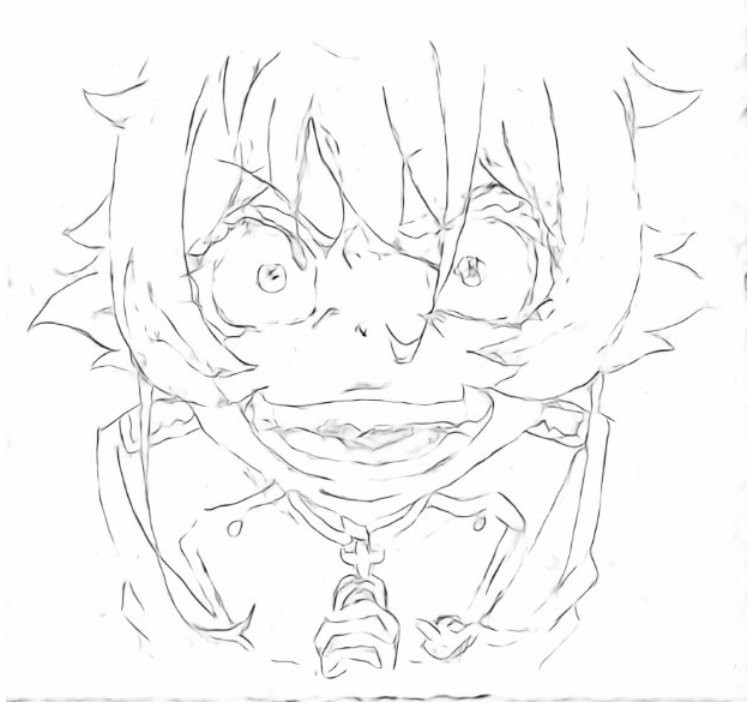
02-4 Dataset generation (7/9)

Result of 1 vs 2: The importance of manual noise (epoch=300)

Regular cartoon image as test data may result in huge amounts of background noises. So it's still not feasible to use common cartoon image as data to train this model today.

02-4 Dataset generation (8/9)

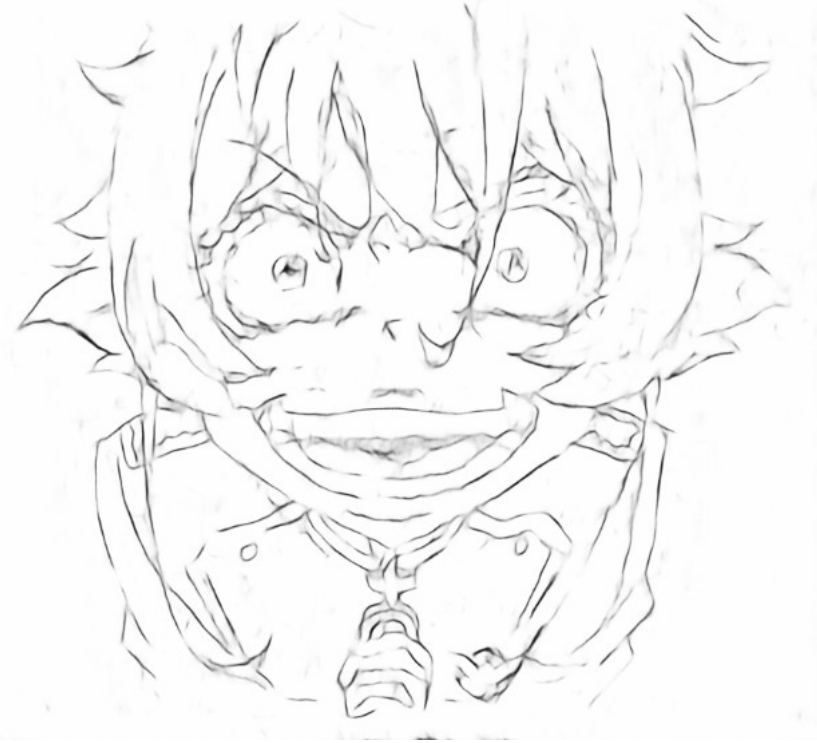
Result of 3 vs 4: The importance of limited noise (epoch=300)



Though generally similar, the result using high-slur has shallower color and shows plenty of grey piles & clusters in many arts of the images. If only use slur images as dataset, the result may get poorer. However in long run the result may become more similar. So the limited noise make some but not much sense.

02-4 Dataset generation (9/9)

Result of 2 vs 5: The importance of patches (epoch=100)



The benefit of using patches is generally get rid of large blank gaps. The result shows the patch may accelerate training to a great extent.

03

Analysis

The work we have done during reproductions.

03 Analysis (1/1)

We've generally reproduced the result of the original paper.

It's a pity that we don't have much time to create more detailing and fruitful dataset. Though the result is acceptable, various reasons including using "fake" groundtruth, short training time and the feature of dataset spark the inability to deal with dense lines and some details.

However the effect of the dataset & patch & limited noise are all shown clearly by the experiment.

04

Discussions

Some further discussions

04 Discussions (1/2)

As we've expected, though the loss map make some sense, the machine learning phase of the author's method has lots of space for improvement (the existence of new models can prove this as well).

However the dataset construction seems to be tedious but pioneering. The comparison experiments have proven that both manual and low noise are important to spark splendid result. And as author have mentioned in the paper, the performance of the model greatly depend on the dataset.

04 Discussions (2/2)

Some future improvement path we've thought:

1) New machine-learning method based on lately models, e.g. SwinTransformer
2) Vectorization methods [Our blind field, attempted and failed]