CS766 project midterm
Generating Chinese Handwriting

Pei-Chieh Lo, Yi Lyu, and Chang-Yen Tseng

{*plo7,ylyu76,ctseng27*}*@wisc.edu*

1. Introduction & Original Proposal

In our proposal, we proposed the idea of generating handwritten Chinese characters. The original intent is to train a neural network model using two sets of input, pictures of printed Chinese characters and pictures of handwritten Chinese characters from the same person. As a result, the model should take a printed Chinese character as an input and produce the same character but with a handwritten style.

2. Dataset

Instead of collecting real human handwriting, which can take a huge amount of labor, we choose to prepare our dataset using existing computer font that mimics handwriting. We also select two printed-style font as we believe transferring style between those should be easier and will be valuable when we are implementing our own model.

3. Domain Specific models

We started by testing implementations that are targeted specially to generate Chinese characters, unfortunately, we are met by some major obstacles primarily due to lack of compatibility and lack of documentations.

3.1. DG-font [5]

This is one of the state of the art research that we mentioned in our proposal. However, after spending lots of time setting up, we cannot get their implementation to work. It is perhaps due to their dependency requiring a rather specific set of hardware that we do not have access to. Nevertheless, their paper provides some key insight that may be helpful for our own implementation. For example, their loss function capture not only style loss but also content consistent lost, which makes sure the network will not produce a character with a similar style but lost strokes that make the character unrecognizable.

3.2. zi2zi [1]

We are able to run their code but the results are unrecognizable (see Figure 1). Consider the lack of documentation (since it is not an academic paper) and that the implementation is not maintained anymore, we decided to move on.

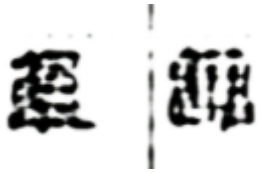


Fig. 1. zi2zi result

4. General style transferring models

Due to those previous issues, we then tried some general style transferring models that are more well-established and have more resources and guides.

4.1. pix2pix [4]

Pix2pix is a really popular model for style transferring and can accomplish incredible tasks such as converting day to night and satellite images to maps. In our evaluation, pix2pix performed quite well in our printed style to printed style test, however, it cannot provide readable output in the printed style to handwritten style case. (Table 1) This

is perhaps because pix2pix, as its name suggests, generally requires the destination pixel have some relationship with the source pixel that is at the same position. Since handwriting is slightly tilted, it does not have this property, therefore greatly impacting pix2pix's performance.

4.2. CycleGAN [2, 9]

CycleGAN is another popular model, unlike pix2pix, it does not require the pairing of the input. Instead, it takes a set of images from style A and another set from style B (those set doesn't necessarily need to overlap) and can perform style transfer either from A to B or from B to A. They achieve this by enforcing cycle consistency. In our evaluation, CycleGAN performs perfectly in the print-to-print style transformation. In the print-to-handwrite case, it can often produce a similar style and readable character. However, important strokes in some characters are lost, sometimes making them unrecognizable. (Table 1) This is likely due to CycleGAN's poor geometric transformation performance and is the main challenge we have to solve.

Nevertheless, CycleGAN performed well and there are many materials and guides on it, so, we decided to first implement CycleGAN and add domain-specific optimization on it.

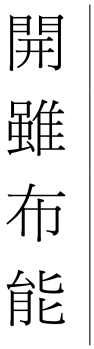
Source	Pix2pix Style1	CycleGAN Style1	Ground Truth Style1	Pix2pix Style2	CycleGAN Style2	Ground Truth Style2
						

Table 1. Result of pix2pix and CycleGAN. Red circle shows the important strokes that are missing

5. Current Progress & Future Work

We are in the progress of implementing our baseline CycleGAN. After implementing and testing using the original dataset in the paper, we will try to test some of the following ideas to improve the output.

- Encoding stroke data [7]
  Providing stroke data will certainly improve the result. The main concern is that gathering stroke data may be a difficult and labor-intensive task. Furthermore, it will be impossible to generate a character that we do not have stroke data.
- Blur the image a little [3]
  Since CycleGAN and pix2pix do not perform well if geometric changes are required, some researchers blurred the dataset so that the geometric changes are not as significant and proved to have improved the result.
- Introduce pre-trained external OCR in our discriminator
  While adding an external model may have defeated the purpose of GAN in its original definition, it may be helpful in our case. After all, our goal is to produce a recognizable output.
- Separating style and content model [5, 8]
  State-of-the-arts seem to use the method a lot. It is easy to see why this method work well, however, this method is very new and therefore may be difficult to implement due to a lack of guides.

6. Responding to the comments of the original proposal

6.1. Evaluation

Obviously, we are still in the progress of implementing are exploring various techniques, so we have yet to discuss evaluation too much. However, we can potentially do a survey for evaluation like Xu et al. [6] did in their research. We can potentially also use the targeted font to write a paragraph and substitute some characters with those we generated, and see if people can identify them.

6.2. Real-time demo

We think it is currently impossible to provide a real-time demo where the user submits just a couple of their handwriting and lets the application train and produce new characters. This is because current models still require tons of data and a long training time. However, it should be easy to pre-train some styles and make a real-time demo where the user types in some characters, and the application can show the character in various styles.

7. Revised Timetable

Date	Goal
04/06	Finish Baseline CycleGAN
04/13	Implement one of the optimization ideas
04/20	Implement another one of the optimization ideas
04/27	Survey & Evaluation
05/03	Final Presentation
05/06	Finish Project Website

References

1. zi2zi: Master chinese calligraphy with conditional adversarial networks.
2. Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. Generating handwritten chinese characters using cyclegan. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 199–207. IEEE, 2018.
3. fendaq. cascaded-pix2pix—chinese-handwriting-generator-, 11 2021.
4. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.
5. Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. Dg-font: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5130–5140, 2021.
6. Songhua Xu, Tao Jin, Hao Jiang, and Francis CM Lau. Automatic generation of personal chinese handwriting by capturing the characteristics of personal handwriting. In *Twenty-First IAAI Conference*, 2009.
7. Jinshan Zeng, Qi Chen, Yunxin Liu, Mingwen Wang, and Yuan Yao. Strokegan: Reducing mode collapse in chinese font generation via stroke encoding. In *proceedings of AAAI*, volume 3, 2021.
8. Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8447–8455, 2018.
9. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.