RYAN XIA - CLASS OF 2022 - IW WRITE UP

GENERATING ART FROM ZUAN SCRIPT: STUDYING CHINESE CALLIGRAPHY WITH RECURRENT NEURAL NETWORKS

ABSTRACT

Chinese seal script characters remain one of the most complex and beautiful forms I have seen and remained thoroughly interested in. From stroke-order to its ordered but chaotic structure, I thought that it would be very interesting to see how one could find patterns and generate new forms based off of these seal script characters. Using Generative Adversarial Networks, Variable Autoencoders, and then Recurrent Neural Networks, I researched, constructed, and generated new seal script forms using a batch of 128 seal script characters.

INTRODUCTION

Chinese seal script is a form of calligraphy during the Qin Dynasty and served as the formal script for any document during that time period. Evolving from bronze script during the Zhou Dynasty, seal script features narrow lines and complicated, square-bounded forms. This would later evolve into an engraving and decorative script used on porcelain, chopsticks, and urns. Furthermore, it would become embedded in square or round seals that marked ownership or authorship of literature and paintings.

Much like most ancient scripts, the characters mean nothing to the average person. This gives a good opportunity to attempt to deceive the average person with computer generated seal script forms. Just like faces or paintings, the idea would be that there are a key, consistent characteristics present that make an imitation convincing. Thus, the goal then becomes to find and define these characteristics such that an algorithm may generate convincing forgeries of actual Chinese Seal Script characters.

Interestingly, there have been many examples of creating forgeries of faces, paintings, and other images using a variety of methods. Furthermore, there are a variety of means used in identifying key characteristics of an image, all of which we will briefly discuss.

PAST WORK

We will discuss three methods of characteristic identification and image generation. Specifically, we will talk about General Adversarial Networks, Recurrent Neural Networks, and Variable Autoencoders and their various applications and downfalls to the goal of the project.

General Adversarial Networks are generative models that rely on two parts: the generator and the discriminator.[1] The generator focuses on generating images and the discriminator tries to differentiate the fake images from the generator from the training data set.[2] There have been several projects done using the GANs structure, including projects generating faces of celebrities by NVIDIA[3] or projects generating abstract paintings such as an example by Varun Salian[4]. These projects, however, also reveal the current limitations of GANs in what it can and cannot generate. Much of the training set needs to be of a similar structure - faces, paintings, boats, trucks, etc. In other words, the discriminator needs to be able to learn specific identifying traits of the entire training set. For example, that eyes don't go on the forehead and noses are in the center of the face. There are a few ways to manipulate this, which can be seen through the two projects mentioned above.

First, is the idea of image curation so that the training set is consistent. NVIDIA's GAN was trained on celebrity red carpet photoshoots, which often yielded very similar results.[5] There were highlights from flashes on the cheekbones, lighting was similar, and face orientation tended to be the same. As a result, the network was able to converge in its adversarial training to produce good results. Furthermore, with faces, there is a lower sensitivity to pixel specific information as the key characteristics matter more than the simple face skin tone. So using high resolution images work well in this case, whereas in some cases, it isn't as desirable.

Second is the idea of abstraction. In Varun Salian's example on generating artworks using GANs, he downscales the images to 64x64 pixels for better results.[6] There are a few reasons for this. First of all, it lowers the specificity of each image. If the paintings were high resolution, it would be difficult for the discriminator to converge as the data set may be too complex.[7] Second, it allows for less precise results which would seem to be abstracted paintings as opposed to unconvincing, larger generated images.[8] Thus, we can see that GANs tend to be very demanding and reliant on its training set in order to generate convincing results and tend to suffer from difficulty in converging on more complex datasets.

Variable Autoencoders and Supervised Neural Networks , on the other hand, are mentioned here due to their prevalence used in discriminative/classifier models. Variable Autoencoders are often used in training classifiers via unsupervised learning, while Supervised Neural Networks often accomplish the same task, just via supervised learning. In a sense, they would be useful in

[1] Overview of gan structure | generative adversarial networks. (n.d.).
[2] Overview of gan structure | generative adversarial networks. (n.d.).
[3] Karras, T., Aila, T., Laine, S. (2018, April 30). Progressive Growing of GANs for Improved Quality, Stability, and Variation
[4] Salian, V. (2020, July 03). Generating art with gans.
[5] Karras, T., Aila, T., Laine, S. (2018, April 30). Progressive Growing of GANs for Improved Quality, Stability, and Variation
[6] Salian, V. (2020, July 03). Generating art with gans.
[7] Ibid.
[8] Ibid.

creating networks to identify key features of Chinese Seal Script. However, the reality is that GANs are forced to identify more features than classifiers due to the need to generate convincing images instead of just identifying and differentiating between images.[9] As a result, they were not used for this project.

Finally, there are **Recurrent Neural Networks**. Simply put, RNNs allow the use of previous outputs as inputs and are often taken advantage of for their ability to take into account historical data.[10] The particular neural network of interest is Sketch RNN, which was originally used to generate drawings of different objects from stroke-order data from online drawing games. Since then, it has been used in a variety of applications, including generating Chinese characters.[11]

Recurrent Neural Networks work particularly well with generating Chinese and Chinese-related characters due to its ability to take into account historical data and maintain consistent weights across time.[12]

Specifically, **Sketch RNN**, which is a variant of a recurrent network developed by David Ha and Douglas Eck, is a Long-Term Short-Term Memory Mixture Density Network (LSTM-MDN) implementation of an RNN[13]. A good way to understand the workings of Sketch RNN is to think of it as a matrix of related probability distributions. Each stroke if followed by the next stroke according to a probability distribution of the next possible strokes learned from the training set. As such, the generator effectively draws the character instead of synthesizing a field of pixels from a seed we provide like we would do in a GANs approach. The network also functions on a series of weights that determine the endings as well as the uncertainty of each generated image, allowing for further control on how chaotic the output from Sketch RNN can be.

DATA COLLECTION & CLEANING

The data required was simply images of Chinese Seal Script. Unfortunately, this proved more difficult than initially imagined. Through the assistance of the East Asian department of Princeton University, I was able to obtain images of several specimens of Seal Script from various different time periods in several different forms. Specifically, the specimens were from the China Digital Library[14], which also included data on various other scripts from Chinese history.

Once several images were chosen and downloaded, they were imported into Adobe Illustrator for vectorization and cleaning. The reason this became necessary was because much of the specimens were very dirty pictures and scans. They were

[9] Overview of gan structure | generative adversarial networks. (n.d.).
[10] Stanford, U. (n.d.). Recurrent neural networks.
[11] Ha, D. (n.d.). Recurrent net dreams up fake Chinese characters in vector format with TensorFlow.
[12] Stanford, U. (n.d.). Recurrent neural networks.
[13] Ha, D. (n.d.). Recurrent net dreams up fake Chinese characters in vector format with TensorFlow.
[14] 阿帕比 全民数字阅读. (n.d.).

dirty in the sense that the background was murky, text color was not uniform, and the characters were blurry. Thus, cleaning was absolutely necessary for a usable dataset. By using the high-contrast image trace function within illustrator, the characters were traced, set in black and white, and separated into individual files of uniform size for training (256x256). The results, after being imported into the GAN, are included in the figure below.

APPROACH & IMPLEMENTATION | **GANS APPROACH**

The primary goal of the project remains to generate convincing forgeries of Chinese Seal Script using machine learning. The initial approach relied on GANs to create forgeries, this was unfortunately the incorrect way of approach, we will discuss in detail in the following section.

Implementation was simple and straightforward, with two five-layer sequential neural networks as the discriminator and generator. Training and implementation were both done using pytorch. Each convolutional layer was followed by a batch normalization and leaky ReLU to accelerate the training time, at least in theory. Testing was not done to see whether the training was actually faster.

In terms of loss calculation and training, it is important to clarify that the discriminator and generator are effectively training one another. With each iteration, the generator passes its images into the discriminator, which attempts to classify each image as either real or fake. Following this, the real images are then passed through the discriminator, which again classifies the input. The loss from both classification tasks, which happens to be binary cross entropy loss or log loss due to the nature of the task, is combined, and the overall loss is minimized via gradient descent and backpropagation via pytorch. Similarly, the generator is trained on its ability to fool the discriminator and also uses binary cross entropy loss.

Furthermore, to increase the efficiency of the neural network, the design decreases the data complexity by only working with one color channel (grayscale). As a result, the produced images are far more crisp and the training progresses faster than if it were to be using three color channels.

To be clear, this is a fairly simple implementation of a GAN, and faces many challenges when dealing with more complex data, such as that of this project.

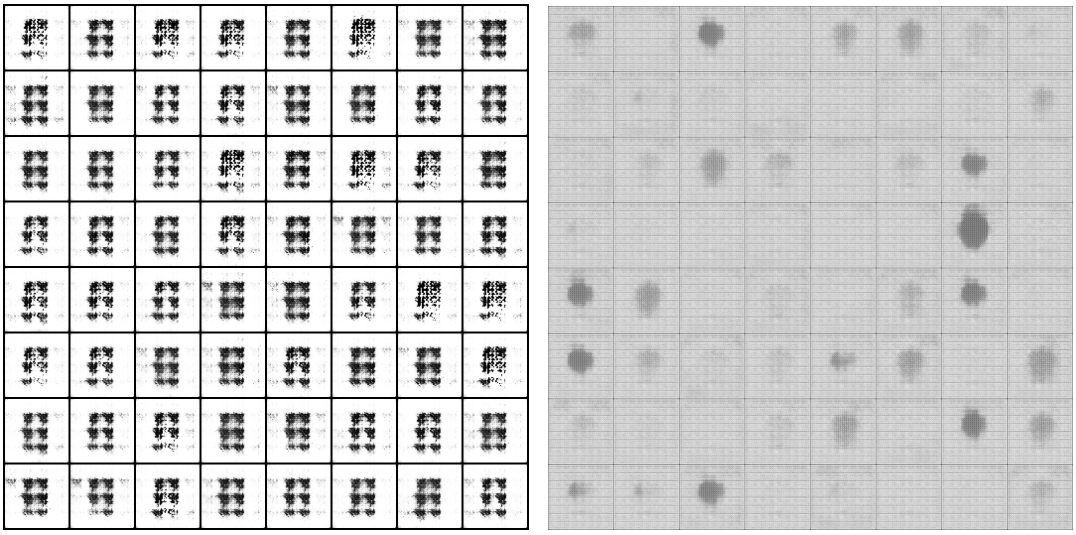
RESULTS | **GANS APPROACH**

The output of the GAN over 150 iterations was less than desired. In essence, the GAN refused to converge generally and instead converged specifically on one or another character. There are a few ways to explain this. One common problem is that, by design, the generator and discriminator agree to settle on one character that both are content with and, as a result, become too specific in their classification criterion. In other words, the generator and discriminator have a training loss that is not sensitive to the actual output of the generator, but only to the discriminator and its criteria. Thus, as long as both agree that a certain output is part of the training set and is "real", then the GAN will converge and produce only one result. This can be combated using Wasserstein loss or an unrolled GAN implementation, both of which work towards moving the discriminator away from stabilizing on one particular criteria[15].

Another possibility is the exact opposite. Non-convergence of a GAN is not uncommon, especially with more complex datasets. What occurs is that the generator is unable to make meaningful progress over small iterations on its output due to the characteristics of the training set to be complex. For this particular issue, one could simply add regularization to the discriminator weights to decrease sensitivity or adding noise to the training set inputs[16]. In both cases, the remedy seeks to assist the discriminator in stabilizing on an input so that the optimization progression can continue.

Simply put, the characters are too specific and too unique relative to one another such that it is difficult for the generator to pick out characteristics that can safely identify all seal script characters in order to successfully lower its loss and fool the discriminator. Interestingly, the results align with this explanation (results on next page). To be clear, there are more methods of remedying GANs convergence failures that could have been implemented.[17] However, given the nature of the training subject, it was deemed unnecessary and inefficient to pursue using GANs any further.

[15] Common problems | generative adversarial networks | google developers. (n.d.).
[16] Ibid.
[17] Ibid.



The figure on the left is a generated image of a 64x64 image size training run after the 150th iteration of adversarial training. A good way to understand these results is that the generator and discriminator have currently arrived at an understanding that there is a white background, and that there is a black pixel mass in the center of the white canvas. Anything that matches these criteria can be considered a seal script character, at least as of the last iteration. The initial reaction was that the characters were over-abstracted due to the lower resolution and, as a result, caused the GAN to be unable to arrive at more specific forgeries.

This was tested by reformatting the generator and discriminator neural net structures to accommodate for 512x512 sized input. The results are presented in the figure to the right, which shows even lower convergence. In both cases, looking at the first 150 runs, the GAN discriminator and generator fluctuated and consistently regressed in their convergence progress over the course of 10 to 20 iterations. This is, again, understandable given the high relative uniqueness Chinese Seal Script characters have. In conclusion, it is safe to say that this behavior of non-specific convergence will persist for most handwriting systems, especially for those that are more complex such as Chinese or Japanese characters.

DATA MANIPULATION | **RNN APPROACH**

A relatively recent development in RNNs is the Sketch RNN, which uses svg ordered-path data to generate drawing of various subjects using probabilistic distributions of various strokes relative to one another. To be clear, this is only a variant of use of Sketch RNN, which has various accepted data inputs. The modules for this particular version of Sketch RNN can be found on GitHub[18].

[18] Apel, U. (n.d.). Kanjivg/kanjivg

Back to the stroke-ordered svg files, the requirement complicates the project. In order to use Sketch RNN, it requires that the Chinese Seal Script character data be converted from high resolution JPG files into stroke based SVG files, which is an arduous task. The process is as follows:

1. Trace/Draw the Seal Script characters in the correct stroke order in Adobe Illustrator
2. Export each artboard as an SVG and open them in a code editor
3. Edit the SVG code to include stroke order and export

After the characters were converted into SVG files, they were uploaded to GitHub as an open source database of stroke-ordered Chinese Seal Script characters. The repository, as it is currently, contains 64 stroke-ordered SVG files of different characters. Unfortunately, not all the stroke orders are correct, and many were extrapolated from my personal knowledge of modernized Chinese characters.

In terms of implementation, the database uses the SVG files from KanjiVG, a separate database for stroke-ordered Japanese Chinese Character SVG files, as references for construction. Due to the construction process of these SVG files, the code is less consistent than that of KanjiVG but will function when used in conjunction with Sketch RNN. The Database will continue to be updated past the completion of this paper and will serve as a resource for others to use in the future. Next steps include standardizing the SVG files for ease of use and further expanding the database.

IMPLEMENTATION | **RNN APPROACH**

Unfortunately, due to version issues with regards to Tensorflow and Magenta[19], the library containing sketch rnn and its related modules, actually implementing a network to work with the stroke-ordered SVG files will be difficult and has not been done at the time of writing this paper. As a result, the next few months will also be spent implementing sketch rnn to work with the newer versions of Tensorflow available.

CONCLUSION

With the database created and filled with 64 characters, the next steps will be filling the database continually with new information. While the initial goal was to generate seal script using GANs, the shift towards using Sketch RNN to create forgeries required a shift in focus of the project towards creating a sustainable database for future use and study of Chinese Seal Script characters. Nonetheless, the project has reached a successful point with a clear goal of enabling vector-based studies of Chinese Seal Script via the public Github repository[20].

[19] Ha, D., & Eck, D. (n.d.). Sketch RNN Repository.
[20] Xia, R. (n.d.). Seal Script Repository.

To further elaborate on the data cleaning and manipulation stages of the project. The paper hides much of the detail with regards to the process due to the arduous nature of the work as well as the inability of such detailed descriptions to contribute significantly to the paper. However, the results and related pieces of code to the project will be included in the public repository for review with proper documentation. This paragraph was included to make clear that much of the work for this project was sunk into the data cleaning and creation process which, unfortunately, does not yield much to show for in terms of words on paper.

ACKNOWLEDGEMENTS

I'd like to thank my Advisor, Brian Kernighan, as well as my Independent Work Seminar classmates for their feedback and advice on the project. More importantly, I would like to thank Brian for reassuring me on the project's viability despite it being a data creation and processing project. I'd also like to thank Joshua Seufert from the East Asian Department for his help in locating and granting access to the database of Chinese Seal Script characters.

SOURCES

Apel, U. (n.d.). Kanjivg/kanjivg. Retrieved May 07, 2021, from https://github.com/KanjiVG/kanjivg/blob/master/kanji/03047.svg?short_path=754ad53

Common problems | generative adversarial networks | google developers. (n.d.). Retrieved May 07, 2021, from https://developers.google.com/machine-learning/gan/problems

Ha, D. (n.d.). Hardmaru/sketch-rnn. Retrieved May 07, 2021, from https://github.com/hardmaru/sketch-rnn/

Ha, D. (n.d.). Recurrent net dreams up fake Chinese characters in vector format with TensorFlow. Retrieved May 05, 2021, from https://blog.otoro.net/2015/12/28/recurrent-net-dreams-up-fake-chinese-characters-in-vector-format-with-tensorflow/

Ha, D., & Eck, D. (n.d.). Sketch RNN Repository. Retrieved May 07, 2021, from https://github.com/magenta/magenta/blob/master/magenta/models/sketch_rnn/README.md

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018, April 30). Progressive Growing of GANs for Improved Quality, Stability, and Variation. Retrieved May 04, 2021, from https://research.nvidia.com/publication/2017-10_Progressive-Growing-of

Overview of gan structure | generative adversarial networks. (n.d.). Retrieved May 04, 2021, from https://developers.google.com/machine-learning/gan/gan_structure

Salian, V. (2020, July 03). Generating art with gans. Retrieved May 04, 2021, from https://blog.jovian.ai/generating-art-with-gans-352ceef3d51f

Stanford, U. (n.d.). Recurrent neural networks. Retrieved May 04, 2021, from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Xia, R. (n.d.). Seal Script Repository. Retrieved May 07, 2021, from https://github.com/RX1334/sealscriptVG

阿帕比 全民数字阅读. (n.d.). Retrieved May 07, 2021, from http://www.apabi.com/