Photo to Doodle Generator Using a GAN

Project Final Report, CS230 Fall'19, Generative Modeling

Talbot Morris-Downing, Manu Agarwal

https://github.com/talbotmd/doodle

Introduction

Many papers and projects have applied deep learning to various problems where the input is a human sketch or doodle. For example retrieving an image based on a human sketch **[1]**, recognizing/classifying the object drawn in a sketch **[3]**, or taking a sketch and creating a realistic image from it **[5]**. However, to our knowledge it seems that very few people, if any, have focused on applying deep learning to go in the other direction, that is taking an image and creating a sketch similar to what a human would create. Certainly algorithms exist for creating drawings from images, but these generally use contours to create much more accurate sketches than what the average person could do. Our early attempts at applying style transfer to these images yielded unappealing results that can easily be distinguished from a human made drawing (below).

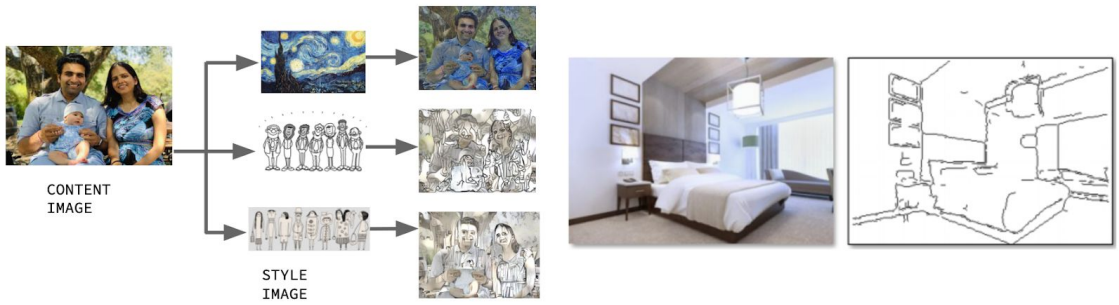

Figure 1: Attempts at making human drawn doodles using style transfer or contour drawing. Results were nonsensical in style transfer when trying to do doodles instead of re-coloration, and results were robotic with machine generated contour drawing.

Related Work

It has been difficult to find work that exactly matches our project, however we were able to find some work that attempted similar or related tasks. In Cai and Song's work **[6]**, they created a model to output images that resemble pencil drawings from an input image. While their process creates images that are very visually appealing, they are certainly beyond what the average human is capable of drawing. Moreover, their process is essentially style transfer, using the image contours as the content, and a separate hand drawn image for the style, and the model does not really learn anything about the nature of the original images.

On the opposite side of the spectrum, there is a fairly significant amount of work that starts with sketches as an input. A 2017 paper by David Ha and Douglas Eck, A Neural Representation of Sketch Drawings **[3]**, uses recurrent neural networks to take human sketches and reconstruct them by encoding and decoding the images. Though using an RNN did not make as much

sense in our case as that work was using the lines drawn as the input and our work used a single image as the input, the process of encoding the image down to a simpler representation seemed interesting, it was also used in the image-to-image translation paper **[5]**. The image-to-image work attempts to do essentially the same problem that we are working on, but in the opposite direction. They take human drawn images, or simplified representations of the desired output, and output realistic images, whereas we input an image, and desire a simplified representation that appears hand drawn as an output.
We found it interesting that both of these papers employed an encoder/decoder network to extract some underlying representation of the data, and we ultimately ended up applying this strategy in our own work as well with fairly positive results.

Dataset

The dataset we are using is the sketchy dataset **[2]**, where researchers have taken sketches from crowd workers, who are tasked to sketch the object in the photo shown. There are 125 categories of objects, 100 photos per category, and 5+ sketches per photo. Overall we have more than 75000 photo+sketch pairs to train with.

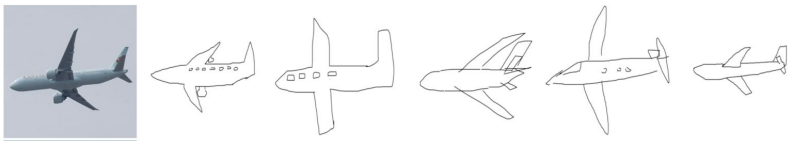

Figure 2: Example of doodles in the sketchy dataset. Category: airplane, one photo and 5 sketches for photo shown.

Data Augmentation: To increase our dataset, we have augmented our it by applying horizontal flip to the photos and sketches for this milestone report.
Note also, that these sketches from the sketchy dataset aren't particularly beautiful, but *are* doodled by hand. Due to this, we are *not* expecting the output of our generator to be pretty, but to look naturally hand drawn and to be a representative of the input photo.

Approach

Our overall goal is to make a doodle generator (***Hp2d***) whose output is a) reminiscent of a hand drawn doodle, and b) represents the input photo. Our approach was to use GANs to simultaneously train an ***Hp2d*** generator and one or two discriminators. The first discriminator (***Dh***) was meant to discriminate a human drawn doodle from one drawn by the generator. The other discriminator (***Dm***) was meant to check if the doodle matches the picture. Initially we tried using two discriminators, but later began using only the discriminator that determined whether or not the image and doodle pair matched. We went through several iterations of network architectures for these models. We started with a fully connected 8 layer neural network. This did not yield very promising results, so we quickly moved on to implementing the generator and discriminators using a series of convolutional layers with leakyReLU activation functions for most layers and sigmoid activation functions for the final layers. Early tests seemed promising, so we iterated on these for the rest of the project, experimentally increasing and decreasing the

complexity of the network trying for better results.
In **[3]**, the authors used an RNN to encode human sketches, and then decoded them to get an output that resembled the input, and an encoding that captured the essence of the input. We attempted a similar strategy of passing the images through a series of CNN layers which reduced the dimensionality of the data, before decoding it into our output. We found that encoding and decoding in this way caused the network to lose a lot of information about the system, and lead to outputs that did not really resemble anything. To help with this issue, we passed the outputs of the encoding layers ahead to the decoding layers with the same dimensions, similar to the residual layers discussed in class. This can be seen more clearly on the poster.

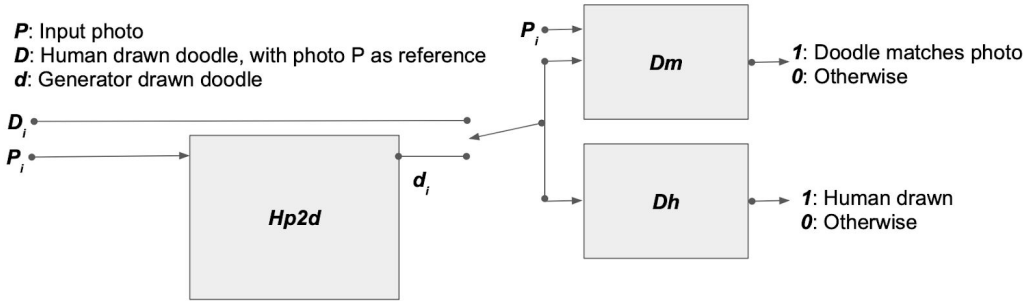

Figure 3: Our original network model structure, one generator and two discriminators.

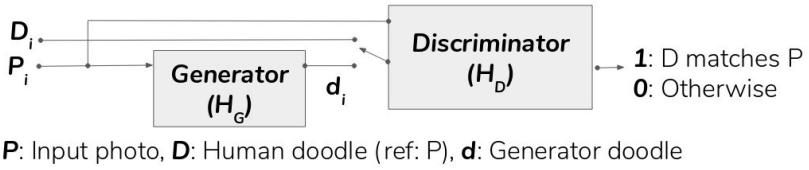

Figure 4: Our revised model structure, one generator and one discriminator

Cost Functions: We used the following cost functions for training the generator and the discriminator (revised model).

$$J_{H_D} = -\frac{1}{m_{human}} \sum_{i=1}^{m_{human}} log(H_D(P_i, D_i)) - \frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} log(1 - H_D(P_i, H_G(P_i)))$$

$$J_{H_G} = -\frac{1}{m_{gen}} \sum_{i=1}^{m_{gen}} log(H_D(P_i, H_G(P_i)))$$

Experimental Results
While the initial results from the fully connected network were perhaps not very encouraging, the results of the cnn architectures suggested we were moving in the right direction. It did not closely resemble the input image to any degree, but certainly it was learning to draw lines.

Figure 5: Example output from our initial cnn test

For the midterm report, several models were tested with varying results. In the most successful case, the GANs were trained on 10,000 image/sketch pairs for 150 epochs. Below are some examples from this test of doodles generated by the generator and performance of the discriminators on images from the test set.

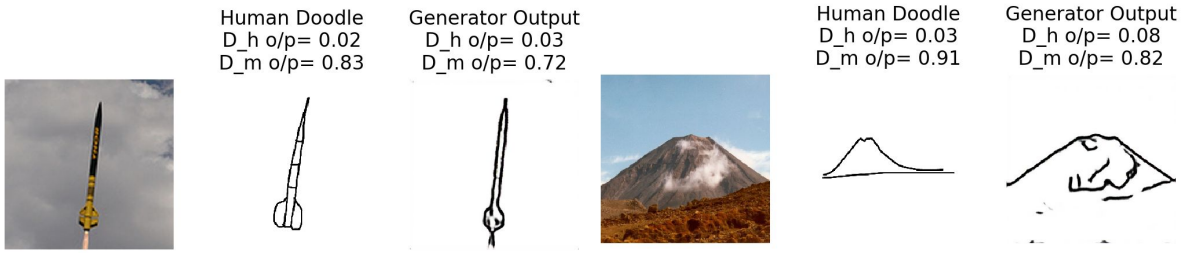

Figure 6: Examples of images from the generator..

Following the midterm report, the improvements and changes that we have made have mostly revolved around trying to improve our convolutional model. We removed the discriminator that was trying to determine if the output was human or computer generated, with the hope that this task could really be performed by the same discriminator that was comparing the input and output images to see if they matched (figure 4).

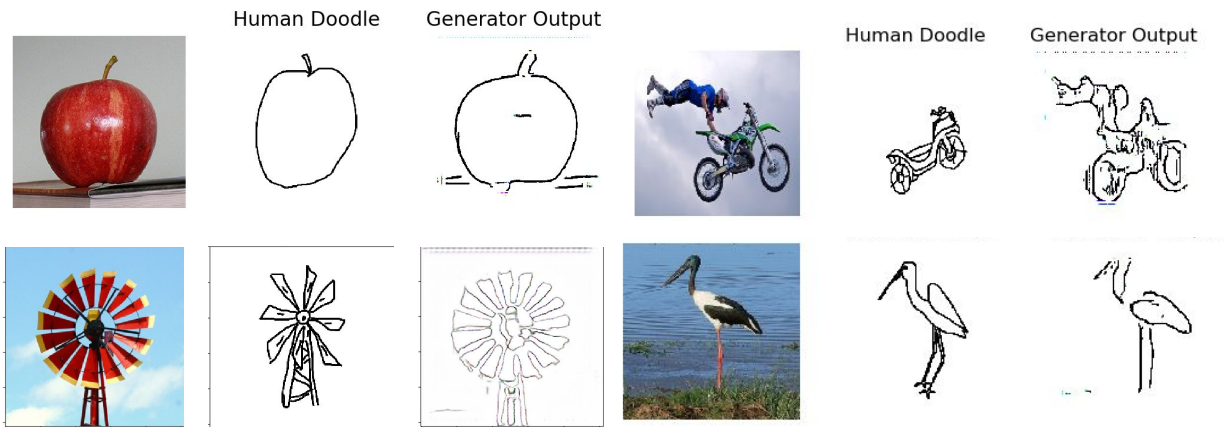

Figure 7: Examples input image, human sketch, and generator output

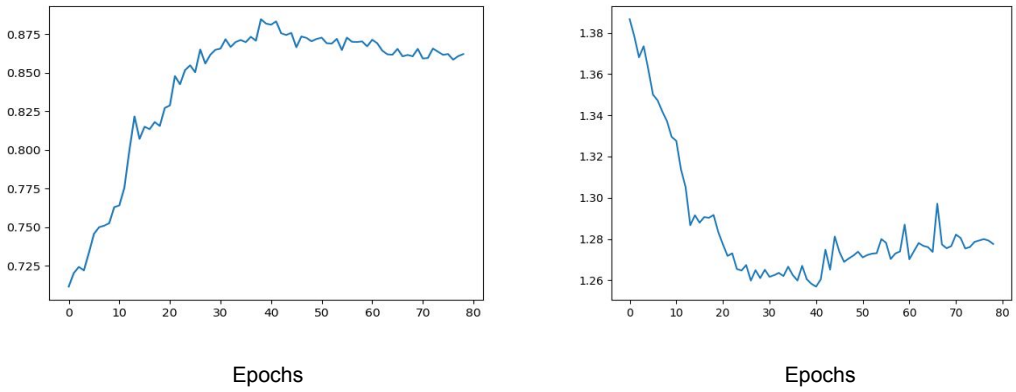

Figure 8: Generator loss (left), Discriminator loss (right), over 80 epochs

In figure 8, the loss for the discriminator can be seen quickly decreasing, which matches a sharp rise in the cost of the generator, followed by the generator adapting and learning to produce outputs that the discriminator has more difficulty distinguishing from human drawings. It seems that the generator is mostly focussing on the contours of the image, rather than learning any kind of understanding of what it is drawing. Given how poorly the input images and sketches match up, it is actually very impressive that the model learned to perform this well. If we had included more information about what object each image contained in the training, it is possible the model could have come up with a simpler representation for the object type.

Future Work

At this point, as mentioned before, the generator still seems to be mainly identifying the contours of the image, and using those to create its output. While this often produces identifiable results, in order to get better performance out of the generator and discriminators on training set, and also to help it generalize, we would suggest a few changes. The network still has difficulty distinguishing between the foreground and background of an image, so using software to artificially extract the foreground before computing the doodle would make it much easier to draw. Additionally, as it seems that the model has really just learned how to draw contours, it would probably benefit significantly from having more information about the objects in the scene it is supposed to be drawing. Using YOLO or some other object recognition software to precompute what should be drawn and where it is in the scene would allow the model to be less general, and have a better understanding of what it is drawing.

Contributions

Manu came up with the model structure, a generator trying to fool two discriminators, cost functions, and built the first fully connected neural network implementation, as well as an encoding based triplet loss implementation that we did not have enough time to fully test. Talbot worked mainly on creating and iterating on the CNN architecture, and setting up/training

on AWS.

References

1. Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The sketchy database: learning to retrieve badly drawn bunnies. ACM Trans. Graph. 35, 4, Article 119 (July 2016), 12 pages. DOI: https://doi-org.stanford.idm.oclc.org/10.1145/2897824.2925954
2. Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. Sketchy Database. http://sketchy.eye.gatech.edu/
3. David Ha, Douglas Eck. 2017. A Neural Representation of Sketch Drawings. https://arxiv.org/abs/1704.03477
4. David Ha, Douglas Eck. 2017. QuickDraw Dataset. https://github.com/googlecreativelab/quickdraw-dataset
5. Isola, Phillip and Zhu, Jun-Yan and Zhou, Tinghui and Efros, Alexei A. 2016. Image-to-Image Translation with Conditional Adversarial Network. https://arxiv.org/abs/1611.07004
6. Xiuxia Cai, Bin Song. Image Based Pencil Drawings Synthesized Using Convolutional Neural Network Feature Maps. https://link.springer.com/article/10.1007%2Fs00138-018-0906-2