# SKETCHIFY: A DETAIL CONSISTENT IMAGE TO DOODLE CONVERTER USING DEEP GENERATIVE NETWORKS

*Prashant Ramnani, Satyam Porwal, Robin Babu, Kousshik Raj*

Indian Institute of Technology, Kharagpur

## ABSTRACT

Deep networks have been generating satisfactory results in a wide range of applications, with a special focus on data extrapolation. But there are a lot of tasks which require data abstraction as well, like, summarizing data, ignoring less important details from images, etc. In this paper, we focus on using deep networks and generative algorithms for abstracting data without losing any of the key points present in it. We present Sketchify, an image to doodle converter, which converts the given input image to its corresponding doodle using deep generative networks, without any inconsistency between the input image and output doodle. Our algorithm also tries to match the major image details, to not lose any of its important characteristics.

***Index Terms*—** Sketch, Doodle, Image Abstraction, Recurrent Neural Networks, Generative Algorithms

## 1. INTRODUCTION

Abstraction plays a vital role in our day to day lives. For example, while reading an article, we subconsciously retrieve the key points in it and store it in our mind. Another example would be the way humans process the surrounding through the visual input we receive through our eyes. We do not particularly try to remember every minute detail we see. Rather we try to remember the various major aspects to it when we do not have any particular focus. In this paper, we present an algorithm, Sketchify, which tries to imitate the process through which humans perceive their surroundings through image abstraction.

As the era of data trudges forward, similar algorithms that focus on interpolation of data, will gain their importance owing to the fact that data is ever increasing and data processing speed has hit a saturation. As different types of deep networks crop up, studies show that, although, they can rival, if not surpass, humans in their efficiency in visual perception, the abstractions learned by the networks drastically differ from that of what is learned by humans.

Thanks to Prof. Abir Das for his guidance.

A study [1] shows that deep networks correctly classify objects when only high-frequency components of the image are preserved but fail completely when only low frequency components are preserved (the low frequency version of image is a blurred version of that image).

Still, we try to create an analogy between the way humans abstract an image they see and the way Sketchify tries to abstract the given image and converting it into a sketch with reference to the image in Fig. 1. Fig. 1 tries to show how a person only remembers a subset of details when faced with the task of reproducing the image of a given scene without reference.

When we first look at the image, we try to figure out the key characteristics of it, which in this case, turns out to be the cake, the toy puppy and the writing. On further scrutiny, we might include the paw prints, and the enclosing box. When trying to recreate the image as a sketch, we attempt to do it by starting with the things we deem as the most important and further proceed on with the less important ones until we reach a point where we ignore certain things. From Fig. 1, we can see that certain paw prints, the box, some of the writings, and the finer details of the puppy have been ignored in the output sketch, but the resulting sketch still almost conveys the same intent as the original image.

Sketchify also follows a similar process. Initially it identifies the various objects in the images and classifies them. Then it converts the objects which it considers important into its corresponding sketch and ignores the rest.

## 2. RELATED WORKS

There is a long history of work related to algorithms that mimic painters. One such work is Portrait Drawing by Paul the Robot [2], where an underlying algorithm controlling a mechanical robot arm sketches lines on a canvas with a programmable artistic style to mimic a given digitized portrait of a person.

Reinforcement Learning based-approaches [3] have also been developed to discover a set of paint brush strokes that can best represent a given input photograph. A work by David Ha et al. [4] proposes a recurrent neural network (RNN) [5] able to construct stroke-based drawings of common objects.


(a) Image to be sketched (b) Manual sketch

**Fig. 1**. Manual reproduction of an image with a lot of details into an abstract sketch

Lamb et al. [6] introduced the task of image abstraction by correlating an image to its corresponding sketch through learning with the aid of deep neural networks. They carried out experiments with various tasks like Rotation Prediction, Virtual Adversarial Training, Adversarial Domain Adaptation, with the aim of matching an unlabelled sketch to the most similar image to the sketch's source.

Another related work by Peng Xu et al. [7], focuses on learning meaningful representations of free-hand sketches by proposing a new representation of sketches as multiple sparsely connected graphs using Graph Neural Network (GNN) [8].

Furthermore, the data extrapolation corresponding to the data abstraction of image to sketch, the sketch based image synthesis, has also received its fair share of attention because of its tremendous applications.

There also has been an increasing interest in the usage of deep convolutional neural networks (dCNN) for more generative tasks. The works of Gatys et al. [9] use these dCNNs to transform the correlations of texture features of the "style" image within each layer of the network into a set of Gram matrices as well as to capture the high-level content of the "content" image. They use the VGG-19 network [10], which is trained only on images from ImageNet.

While all these works mainly focus on various tasks related to image-sketch correlation, like learning from sketch, image synthesis, etc., there is yet to be a state-of-the-art algorithm for abstracting an image into an sketch that best represents it.

## 3. METHODOLOGY

### 3.1. Dataset

We primarily use the *Quick, Draw!* [11] dataset for training our Sketch-RNN, which consists of vector drawings obtained from *Quick, Draw!*, an online game where a player is asked to draw an object of a particular category within 20 seconds. The categories are the hundreds of common objects which we encounter in our daily life.

The data format used to represent the sketch is a set of pen stroke actions. Each binary pen stroke event corresponds to multi-state event, consisting of 5 elements $(\Delta x, \Delta y, p_1, p_2, p_3)$, where $\Delta x$ and $\Delta y$ correspond to the offset distance in x and y from previous state. The last 3 states represent the state of the pen, $p_1$ corresponds to pen is drawing, $p_2$ corresponds to the pen is lifted and $p_3$ corresponds to drawing has ended.

The data from *Quick, Draw!* expands daily and every so often new classes are added to the data set. Currently the data set consists of 345 classes, out of which a small list is listed in Table 1.

| | | | |
|---|---|---|---|
| Airplane | Birthday Cake | Candle | Cloud |
| Cat | Dog | Eye | Fan |
| Flower | Grass | Hat | Ice Cream |
| Moon | Pencil | Pizza | River |
| Shoe | Sun | Table | Umbrella |
| | Water-melon | Wine Glass | |

**Table 1**. A small subset of the 345 different classes in the QuickDraw dataset

We used the COCO dataset [12] for training the object detection module. Common Object Context (COCO) is a well known standard dataset used for the training object detection modules. It consists of around 330,000 images, and roughly 1.5 million instances of objects belonging to 80 different object categories and 91 stuff categories.

### 3.2. Object Detection

The input image is first passed through a Faster-RCNN [13] model trained over COCO dataset. Our aim at this step is to detect and classify the various entities in the image so that we can extract all the major characteristics of the image. The model generates a bounding box around the detected object and labels them with the class which it thinks matches the object best. Fig. 2b shows the result of the model trained over COCO dataset on Fig. 2a. The result shows a dog and a man labelled with more than 90% accuracy.

These labels and positions for each entity are then used to generate the corresponding sketches for each instance in a simple manner by selecting images that resemble the object closely. Since the *Quick, Draw!* dataset is used to train the Sketch-RNN and also used for the initial instance replacement, there might be inconsistency in the classes as *Quick, Draw!* and COCO have different classes or have different labels for the same class. Hence, we create a mapping between the two sets of classes using *Synset* and *Similarity*. Table 2

shows a subset of the mapping generated between the two different set of classes.

| *Quick, Draw!* | COCO |
|---|---|
| tv | television |
| surfboard | skateboard |
| sports ball | soccer ball |

**Table 2**. A small subset of the mapping between two different sets of classes in *Quick, Draw!* and COCO dataset

(a) Original Image (b) Object Detection Result

**Fig. 2**. Object detection using a Faster-RCNN model

### 3.3. Sketch-RNN

#### *3.3.1. Motivation*

As our goal is to create a simple sketch of the given image, initially we chose to find a sketch resembling the individual objects in the image based on the labels generated by the object detection module. This provides a low level pixel based representation of the image. However, this was incomplete and did not provide a sense of uniqueness to objects that might differ even among the same class. This motivated us to create a network that could further refine this low level image representation to be a slightly more improved abstract representation of the image.

Furthermore the objects that were placed in the initial method showed very less signs of influence and were largely independent of other objects giving the impression of a lack of interaction between them.

We created a Generative Recurrent Neural Network that is capable of taking simplistic incomplete cartoon images and fleshing them out to give a more detailed and better cartoon representation of the objects. Passing the image through this generative-RNN made the image more complete and natural.

#### *3.3.2. Architecture*

We are using a generative recurrent neural network model capable of producing sketches of common objects to draw the sketch of the image, with the goal of training a machine to draw and generalize abstract concepts in a manner similar to humans.

We investigate a lower-dimensional vector-based representation inspired by how people draw. The Sketch-RNN model is based on the sequence-to-sequence (seq2seq) auto-encoder framework [14] [15]. It incorporates variational inference and utilizes hyper networks as recurrent neural network cells. The goal of the seq2seq auto-encoder is to train a network to encode an input sequence into a latent vector, and from this latent vector it reconstructs an output sequence using a decoder that replicates the input sequence as closely as possible.

In the model, noise is added to the latent vector by inducing noise into the communication channel between the encoder and the decoder, hence the model will not able to reproduce the input sketch exactly and must learn to capture the essence of the sketch as a noisy latent vector. This is depicted in Fig. 3. The decoder takes this latent vector and produces a sequence of motor actions used to construct a new sketch.

For our purpose, the model was trained on the *Quick, Draw!* dataset for certain categories and an SVG for each category can be generated from the trained data (conditional generation). For each category, the model generates new sketches by not just simply copying the input sequence.

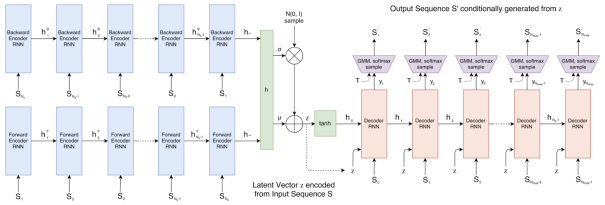


**Fig. 3**. Schematic of RNN

#### *3.3.3. Loss Function*

Our training procedure follows the approach of the Variational Autoencoder [16], where the loss function is the sum of two terms: the Reconstruction Loss, $L_R$, and the Kullback-Leibler Divergence Loss, $L_{KL}$.

$L_R$ is further composed of the sum of the log loss of the offset terms ($\Delta$x, $\Delta$y), $L_s$, and the log loss of the pen state terms $(p1, p2, p3)$, $L_p$.

$$L_s = -\frac{1}{N_{\max}} \sum_{i=1}^{N_s} \log \Big( \sum_{j=1}^{M} \Pi_{j,i}\, \mathcal{N}(\Delta x_i, \Delta y_i \mid \mu_{x,j,i}, \mu_{y,j,i}, \sigma_{x,j,i}, \sigma_{y,j,i}, \rho_{xy,j,i}) \Big)$$

$$L_p = \frac{-1}{N_{max}} \sum_{i=1}^{N_s} \sum_{k=1}^{3} p_{k,i} \log(q_{k,i})$$

$$L_R = L_s + L_p$$

The Kullback-Leibler $(KL)$ divergence loss term measures the difference between the distribution of our latent vector z, to that of an IID Gaussian vector with zero mean and unit variance. Optimizing for this loss term allows us to minimize this difference.

$$L_{KL} = \frac{-1}{2N_s}(1 + \hat{\sigma} - \mu^2 - \exp(\hat{\sigma}))$$

The loss function is a weighted sum of both the $L_R$ and $L_{KL}$ loss terms.

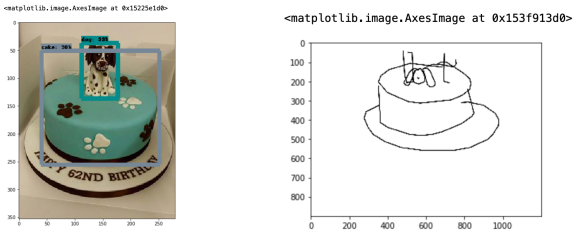
$$Loss = L_R + w_{KL}LKL$$

### 3.4. Final Sketch Generation

We use the object detection results to Sketchify the image by replacing the objects with cartoon drawings of the images in a very simplistic manner. The final integration proceeds by passing the bounding box, location and size to the Sketch-RNN model, which is a generative recurrent neural network running in conditional generation mode. Thus, given a low level representation it creates improvements to the images based on the dataset of drawings that it was trained on. It produces a more complete and high level cartoon representation of the object by depicting the interaction between the various entities, thus creating the final sketch.

## 4. RESULTS

Fig. 4 shows the result generated from the model with Fig 1.a as input. Fig. 4a shows the result of object detection, showing the categories, dog and cake. Fig. 4b shows the final result of the model. The result shows that the network has learned the major essence of image, the cake and the dog and tried to produce a complete sketch. The result when compared to Fig. 1b, the human drawn sketch of the same image, shows that the model captured a lot of the high-level details.




(a) Object Detection (b) Final Generated Sketch

**Fig. 4**. Result generated using Sketchify on Fig. 1a

Fig. 5 shows the output of the model when there is complex interaction between the different entities in the image. Similarly, Fig. 6 shows results of Sketchify on an image of a cat. The final sketch generated by model was again passed through the conditional generative-RNN to generate further sketches and the best sketch was selected (Fig. 6b).

## 5. FUTURE WORKS

Free-hand sketch has its domain-unique technical challenges, since it's essentially different from a natural photo. Furthermore, the implementation of Sketchify is quite constrained and can be extended in a few ways:

- The object detection only identifies what category an object belongs to, there is no data available about the orientation and other details.
- The sketches generated by using Sketch-RNN, though better than randomly choosing a drawing from original dataset, still shows scope for improvement, somewhat due to poorly drawn images in the dataset used.

In recent years Generative Adversarial Networks (GANs) [17] have shown significant success in modelling higher dimensional distribution of visual data. One of such research developed a method to generate images from hand-drawn sketches [18]. Models like these would enable many exciting new creative applications in a variety of different directions. They can also serve as a tool to help us improve our understanding of our own creative thought processes.

(a) Original Image (b) Generated Sketch

**Fig. 5**. Result generated using Sketchify

(a) Original Image (b) Final Generated Sketch

**Fig. 6**. Result generated using Sketchify and further passed through conditional generative-RNN

## 6. CONCLUSION

In this work, we came up with a low level strategy to create a simplistic sketch of an image based on the objects present in an image. Here, an inherent assumption we have taken is that the objects are independent of each other. We further go on to improve this methodology by also implementing a generative RNN that is capable of completing images, refining them, thus inducing this dependence among the objects and creating a more 'perfect' cartoon sketch of the image.

## 7. REFERENCES


[1] Haohan Wang, Xindi Wu, Pengcheng Yin, and Eric P. Xing, "High frequency component helps explain the generalization of convolutional neural networks," *CoRR*, vol. abs/1905.13545, 2019.

[2] Patrick Tresset and Frederic Fol Leymarie, "Portrait drawing by paul the robot," *Computers Graphics*, vol. 37, pp. 348–363, 08 2013.

[3] Ning Xie, Hirotaka Hachiya, and Masashi Sugiyama, "Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting," *CoRR*, vol. abs/1206.4634, 2012.

[4] David Ha and Douglas Eck, "A neural representation of sketch drawings," *CoRR*, vol. abs/1704.03477, 2017.

[5] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra, "DRAW: A recurrent neural network for image generation," *CoRR*, vol. abs/1502.04623, 2015.

[6] Alex Lamb, Sherjil Ozair, Vikas Verma, and David Ha, "Sketchtransfer: A challenging new task for exploring detail-invariance and the abstractions learned by deep networks," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 963–972.

[7] Peng Xu, Chaitanya K. Joshi, and Xavier Bresson, "Multi-graph transformer for free-hand sketch recognition," 2019.

[8] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun, "Graph neural networks: A review of methods and applications," *CoRR*, vol. abs/1812.08434, 2018.

[9] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "A neural algorithm of artistic style," *CoRR*, vol. abs/1508.06576, 2015.

[10] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[11] T. Kawashima J. Kim J. Jongejan, H. Rowley and N. Fox-Gieg, "The quick, draw! - a.i. experiment," quickdraw.withgoogle.com, 2016.

[12] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[13] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[15] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[16] Diederik P. Kingma and Max Welling, "An introduction to variational autoencoders," *CoRR*, vol. abs/1906.02691, 2019.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial networks," 2014.

[18] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou, "Sketchycoco: Image generation from freehand scene sketches," 2020.