A Neural Representation of Sketch Drawings

David Ha
Google Brain
hadavid@google.com

Douglas Eck
Google Brain
deck@google.com

Abstract

We present `sketch-rnn`, a recurrent neural network (RNN) able to construct stroke-based drawings of common objects. The model is trained on a dataset of human-drawn images representing many different classes. We outline a framework for conditional and unconditional sketch generation, and describe new robust training methods for generating coherent sketch drawings in a vector format.

1 Introduction

Recently, there have been major advancements in generative modelling of images using neural networks as a generative tool. Generative Adversarial Networks (GANs) [5], Variational Inference (VI) [15], and Autoregressive (AR) [19] models have become popular tools in this fast growing area. Most of the work thus far has been targeted towards modelling low resolution, pixel images. Humans, however, do not understand the world as a grid of pixels, but rather develop abstract concepts to represent what we see. From a young age, we develop the ability to communicate what we see by drawing on paper with a pencil or crayon. In this way we learn to express a sequential, vector representation of an image as a short sequence of strokes. In this paper we investigate an alternative to traditional pixel image modelling approaches, and propose a generative model for vector images.

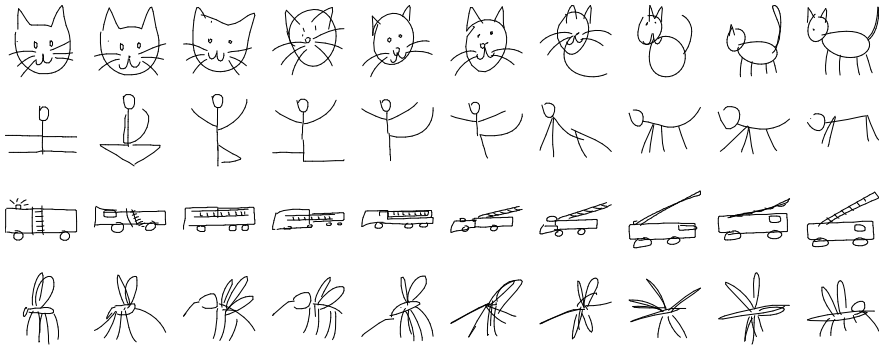

Figure 1: Latent space interpolation of various vector images produced by our model.

Our goal is to train machines to draw and generalize abstract concepts in a manner similar to humans. As a first step towards this goal, we train our model on a dataset of hand-drawn sketches, each represented as a sequence of motor actions controlling a pen: which direction to move, when to lift the pen up, and when to stop drawing. In doing so, we created a model that potentially has many applications, from assisting the creative process of an artist, to helping teach students how to draw.

This paper makes the following contributions: We outline a framework for both unconditional and conditional generation of vector images composed of a sequence of lines. Our recurrent neural network-based generative model is capable of producing sketches of common objects in a vector format. We develop a training procedure unique to vector images to make the training more robust. In

草图绘画的神经表征

大卫·哈
谷歌大脑
hadavid @google .com

道格拉斯·埃克
谷歌大脑
deck @google .com

摘要

我们介绍了sketch - rnn，一种能够构建基于笔画的常见物体绘图的循环神经网络（RNN）。该模型在一个包含许多不同类别的人工绘制图像的数据集上进行训练。我们概述了有条件和无条件的草图生成框架，并描述了一种新的稳健训练方法，用于以矢量格式生成连贯的草图绘制。

1 介绍

最近，使用神经网络作为生成工具对图像进行生成建模取得了重大进展。生成对抗网络（GANs）[5]、变分推断（VI）[15]和自回归（AR）[19]模型已成为这个快速发展领域中流行的工具。迄今为止，大部分工作都集中在对低分辨率像素图像进行建模。然而，人类并不将世界理解为像素网格，而是通过发展抽象概念来表示我们所看到的东西。从小的时候起，我们就通过用铅笔或蜡笔在纸上画画来学会表达图像的顺序、向量表示，将其表示为一系列简短的笔画。在本文中，我们研究了传统像素图像建模方法的替代方案，并提出了一种用于向量图像的生成模型。

图1：由我们的模型生成的各种矢量图像的潜空间插值。

我们的目标是训练机器以类似人类的方式绘制和概括抽象概念。作为实现这一目标的第一步，我们在一个手绘草图数据集上训练我们的模型，每个草图都被表示为一系列控制笔的运动动作：移动方向、何时抬起笔、何时停止绘制。通过这样做，我们创建了一个潜在具有许多应用的模型，从辅助艺术家的创作过程到帮助教学生如何绘画。

本文的贡献如下：我们概述了一个框架，用于无条件和有条件生成由一系列线条组成的矢量图像。我们基于循环神经网络的生成模型能够以矢量格式生成常见物体的草图。我们开发了一种独特的矢量图像训练过程，使训练更加稳健。

the conditional generation model, we explore the latent space developed by the model to represent a vector image. We also discuss potential creative applications of our methodology. We make available a large dataset of hand drawn vector images to encourage further development of generative modelling for vector images, and also release an implementation of our model as an open source project.[1]

2 Related Work

There is a long history of work related to algorithms that mimic painters. One such work is Portrait Drawing by Paul the Robot [23, 25], 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 [25] have been developed to discover a set of paint brush strokes that can best represent a given input photograph. These prior works generally attempt to mimic digitized photographs, rather than develop generative models of vector images.

Neural Network-based approaches have been developed for generative models of images, although the majority of neural network-related research on image generation deal with pixel images [5, 10, 12, 14, 19, 24]. There has been relatively little work done on vector image generation using neural networks. An earlier work [22] makes use of Hidden Markov Models to synthesize lines and curves of a human sketch. More recent work [6] on handwriting generation with Recurrent Neural Networks laid the groundwork for utilizing Mixture Density Networks [1] to generate continuous data points. Recent works of this approach attempted to generate vectorized Kanji characters unconditionally [7] and conditionally [26] by modelling Chinese characters as a sequence of pen stroke actions.

In addition to unconditionally generating sketches, we also explore encoding existing sketches into a latent space of embedding vectors. Previous work [2] outlined a methodology to combine Sequence-to-Sequence models with a Variational Autoencoder to model natural English sentences in latent vector space. A related work [16], utilizes probabilistic program induction, rather than neural networks, to perform one-shot modelling of the Omniglot dataset containing images of symbols.

One of the factors limiting research development in the space of generative vector drawings is the lack of publicly available datasets. Previously, the Sketch dataset [4], consisting of 20K vector sketches, was used to explore feature extraction techniques. A subsequent work, the Sketchy dataset [20], provided 70K vector sketches along with corresponding pixel images for various classes. This allowed for a larger-scale exploration of human sketches. ShadowDraw [17] is an interactive system that predicts what a finished drawing looks like based on a set of incomplete brush strokes from the user while the sketch is being drawn. ShadowDraw used a dataset of 30K raster images combined with extracted vectorized features. In this work, we use a much larger dataset of vector sketches that is made publicly available.

3 Methodology

3.1 Dataset

We constructed `QuickDraw`, a dataset of vector drawings obtained from *Quick, Draw!* [11], an online game where the players are asked to draw objects belonging to a particular object class in less than 20 seconds. `QuickDraw` consists of hundreds of classes of common objects. Each class of `QuickDraw` is a dataset of 70K training samples, in addition to 2.5K validation and 2.5K test samples.

We use a data format that represents a sketch as a set of pen stroke actions. This representation is an extension of the format used in [6]. Our format extends the binary pen stroke event into a multi-state event. In this data format, the initial absolute coordinate of the drawing is located at the origin. A sketch is a list of points, and each point is a vector consisting of 5 elements: $(\Delta x, \Delta y, p_1, p_2, p_3)$. The first two elements are the offset distance in the x and y directions of the pen from the previous point. The last 3 elements represents a binary one-hot vector of 3 possible states. The first pen state, p_1, indicates that the pen is currently touching the paper, and that a line will be drawn connecting the next point with the current point. The second pen state, p_2, indicates that the pen will be lifted from the paper after the current point, and that no line will be drawn next. The final pen state, p_3, indicates that the drawing has ended, and subsequent points, including the current point, will not be rendered.

[1] The code and dataset is available at `https://magenta.tensorflow.org/sketch_rnn`.

在条件生成模型中，我们探索模型开发的潜在空间，以表示矢量图像。我们还讨论了我们方法的潜在创造性应用。我们提供了一个大型手绘矢量图像数据集，以鼓励进一步发展矢量图像的生成建模，并发布了我们的模型实现作为一个开源项目。

2 相关工作

有很长一段时间的工作与模仿画家的算法相关。其中一项工作是由机器人保罗进行的肖像绘画[23，25]，其中一个控制机械机器人手臂的底层算法以可编程的艺术风格在画布上勾勒线条，以模仿给定的数字化人像肖像。基于强化学习的方法[25]已经被开发出来，以发现一组能够最好地代表给定输入照片的画笔笔触。这些先前的工作通常试图模仿数字化照片，而不是开发生成矢量图像的生成模型。

虽然神经网络已经被用于生成图像的生成模型，但大部分与神经网络相关的图像生成研究都是处理像素图像[5，10，12，14，19，24]。在使用神经网络进行矢量图像生成方面的研究相对较少。早期的一项研究[22]利用隐马尔可夫模型合成人类素描的线条和曲线。最近的一项关于手写生成的研究[6]使用循环神经网络为利用混合密度网络[1]生成连续数据点奠定了基础。这种方法的最近研究尝试无条件地生成矢量化的汉字字符[7]，并通过将汉字字符建模为一系列笔画动作来有条件地生成[26]。

除了无条件生成草图之外，我们还探索将现有的草图编码为嵌入向量的潜在空间。之前的工作[2]概述了一种将序列到序列模型与变分自编码器结合起来，在潜在向量空间中建模自然英语句子的方法。一项相关的工作[16]利用概率程序归纳，而不是神经网络，对包含符号图像的Omniglot 数据集进行一次性建模。

在生成矢量图绘画领域，限制研究发展的因素之一是缺乏公开可用的数据集。之前，Sketch 数据集[4]包含了2万个矢量素描，被用于探索特征提取技术。随后的工作，Sketchy 数据集[20]提供了7万个矢量素描以及相应的像素图像，涵盖了各种类别。这使得对人类素描进行了更大规模的探索。ShadowDraw [17]是一个交互式系统，根据用户绘制素描时的一组不完整的画笔笔触，预测最终的绘画效果。ShadowDraw 使用了3万个栅格图像的数据集，结合提取的矢量化特征。在本研究中，我们使用了一个更大的公开可用的矢量素描数据集。

3 方法论

3.1 数据集

我们构建了QuickDraw 数据集，该数据集是从Quick，Draw！[11]中获取的矢量图绘画。Quick，Draw！是一个在线游戏，玩家被要求在不到20 秒的时间内绘制属于特定物体类别的对象。QuickDraw 包含了数百个常见物体类别。每个QuickDraw 类别都包含70K 个训练样本，以及2.5K 个验证样本和2.5K 个测试样本。

我们使用一种数据格式来表示草图，将其表示为一组笔画动作。这种表示是在[6]中使用的格式的扩展。我们的格式将二进制笔画事件扩展为多状态事件。在这种数据格式中，绘图的初始绝对坐标位于原点。草图是一个点的列表，每个点是一个由5个元素组成的向量：(Δx, Δy, p, p, p)。前两个元素是笔从上一个点到当前点在x和y方向上的偏移距离。最后3个元素表示一个二进制的独热向量，有3种可能的状态。第一个笔状态p表示笔当前接触纸张，并且将绘制一条连接下一个点和当前点的线。第二个笔状态p表示在当前点之后将抬起笔，并且下一个点不会绘制线条。最后一个笔状态p表示绘图已结束，并且后续点，包括当前点，将不会被渲染。

[1] 代码和数据集可在https://magenta.tensorflow.org/sketch_rnn上找到。

3.2 Sketch-RNN

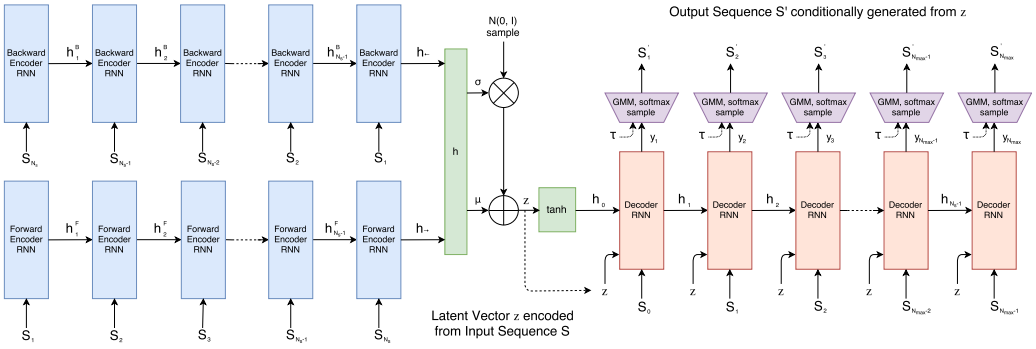

Figure 2: Schematic diagram of `sketch-rnn`.

Our model is a Sequence-to-Sequence Variational Autoencoder (VAE), similar to the architecture described in [2, 15]. Our encoder is a bidirectional RNN [21] that takes in a sketch as an input, and outputs a latent vector of size N_z. Specifically, we feed the sketch sequence, S, and also the same sketch sequence in reverse order, $S_{\texttt{reverse}}$, into two encoding RNNs that make up the bidirectional RNN, to obtain two final hidden states:

$$h_{\rightarrow} = \texttt{encode}_{\rightarrow}(S),\ h_{\leftarrow} = \texttt{encode}_{\leftarrow}(S_{\texttt{reverse}}),\ h = [\ h_{\rightarrow}\ ;\ h_{\leftarrow}\ ] \tag{1}$$

We take this final concatenated hidden state, h, and project it into two vectors μ and $\hat{\sigma}$, each of size N_z, using a fully connected layer. We convert $\hat{\sigma}$ into a non-negative standard deviation parameter σ using an exponential operation. We use μ and σ, along with $\mathcal{N}(0, I)$, a vector of IID Gaussian variables of size N_z, to construct a random vector, $z \in \mathbb{R}^{N_z}$, as in the approach for a VAE [15]:

$$\mu = W_\mu h + b_\mu,\ \hat{\sigma} = W_\sigma h + b_\sigma,\ \sigma = \exp\left(\frac{\hat{\sigma}}{2}\right),\ z = \mu + \sigma \odot \mathcal{N}(0, I) \tag{2}$$

Under this encoding scheme, the latent vector z is not a deterministic output for a given input sketch, but a random vector conditioned on the input sketch.

Our decoder is an autoregressive RNN that samples output sketches conditional on a given latent vector z. The initial hidden states h_0, and optional cell states c_0 (if applicable) of the decoder RNN is the output of a single layer network: $[\ h_0\ ;\ c_0\ ] = \tanh(W_z z + b_z)$

At each step i of the decoder RNN, we feed the previous point, S_{i-1} and the latent vector z in as a concatenated input x_i, where S_0 is defined as $(0, 0, 1, 0, 0)$. The output at each time step are the parameters for a probability distribution of the next data point S_i. In Equation 3, we model $(\Delta x, \Delta y)$ as a Gaussian mixture model (GMM) with M normal distributions as in [1, 6], and (q_1, q_2, q_3) as a categorical distribution to model the ground truth data (p_1, p_2, p_3), where $(q_1 + q_2 + q_3 = 1)$ as done in [7] and [26]. Unlike [6], our generated sequence is conditioned from a latent code z sampled from our encoder, which is trained end-to-end alongside the decoder.

$$p(\Delta x, \Delta y) = \sum_{j=1}^{M} \Pi_j\ \mathcal{N}(\Delta x, \Delta y \mid \mu_{x,j}, \mu_{y,j}, \sigma_{x,j}, \sigma_{y,j}, \rho_{xy,j}),\ \text{where}\ \sum_{j=1}^{M} \Pi_j = 1 \tag{3}$$

$\mathcal{N}(x, y|\mu_x, \mu_y, \sigma_x, \sigma_y, \rho_{xy})$ is the probability distribution function for a bivariate normal distribution. Each of the M bivariate normal distributions consist of five parameters: $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho_{xy})$, where μ_x and μ_y are the means, σ_x and σ_y are the standard deviations, and ρ_{xy} is the correlation parameter of each bivariate normal distribution. An additional vector Π of length M, also a categorical distribution, are the mixture weights of the Gaussian mixture model. Hence the size of the output vector y is $5M + M + 3$, which includes the 3 logits needed to generate (q_1, q_2, q_3).

The next hidden state of the RNN, generated with its forward operation, projects into the output vector y_i using a fully-connected layer:

$$x_i = [\ S_{i-1}\ ;\ z\ ],\ [\ h_i\ ;\ c_i\ ] = \texttt{forward}(x_i,\ [\ h_{i-1}\ ;\ c_{i-1}\ ]),\ y_i = W_y h_i + b_y,\ y_i \in \mathbb{R}^{6M+3} \tag{4}$$

The vector y_i is broken down into the parameters of the probability distribution of the next data point:

$$[\ (\hat{\Pi}_1\ \mu_x\ \mu_y\ \hat{\sigma}_x\ \hat{\sigma}_y\ \hat{\rho}_{xy})_1\ \ldots\ (\hat{\Pi}_1\ \mu_x\ \mu_y\ \hat{\sigma}_x\ \hat{\sigma}_y\ \hat{\rho}_{xy})_M\ (\hat{q}_1\ \hat{q}_2\ \hat{q}_3)\ ] = y_i \tag{5}$$

3.2 素描- RNN

图2： sketch - rnn 的示意图。

我们的模型是一个序列到序列的变分自编码器（VAE），类似于[2, 15]中描述的架构。我们的编码器是一个双向RNN [21]，它以草图作为输入，并输出大小为N的潜在向量。具体来说，我们将草图序列S和相同的反向草图序列S输入到两个组成双向RNN的编码RNN中，以获得两个最终的隐藏状态。

我们将最终连接的隐藏状态h投影到两个大小为N的向量μ和ˆσ中，使用一个全连接层。我们使用指数运算将ˆσ转换为非负标准差参数σ。我们使用μ和σ，以及 $\vec{h}$ $\overleftarrow{h}$ $(0, I)$，一个由独立同分布的高斯向量组成大小为N的变量，构建一个随机向量z $z \in \mathbb{R}^N$ 如同VAE [15]的方法中一样：

$$\mu = W_\mu h + b_\mu, \hat{\sigma} = W_\sigma h + b_\sigma, \sigma = \exp\left(\frac{\hat{\sigma}}{2}\right), z = \mu + \sigma \odot \mathcal{N}(0, I) \quad (2)$$

在这种编码方案下，潜在向量z不是给定输入草图的确定性输出，而是在输入草图的条件下的随机向量。

我们的解码器是一个自回归的循环神经网络，根据给定的潜在向量z生成输出草图样本。解码器RNN的初始隐藏状态h和可选的细胞状态c （如果适用）是单层网络的输出：$[h; c] = \tanh(W_z z + b_z)$

在解码器RNN的每一步i中，我们将前一个点S、潜在向量z作为连接输入x进行输入，其中S被定义为$(0, 0, 1, 0, 0)$。每个时间步的输出是下一个数据点S的概率分布的参数。在公式3中，我们将(Δx, Δy)建模为一个具有M个正态分布的高斯混合模型(GMM)，如[1, 6]所示，将(q, q, q)建模为一个分类分布，以建模真实数据(p, p, p)，其中(q+ q+ q= 1)，如[7]和[26]所做的那样。与[6]不同的是，我们生成的序列是从我们的编码器中采样的潜在代码z进行条件训练的，该编码器与解码器一起进行端到端训练。

$$p(\Delta x, \Delta y) = \sum_{j=1}^{M} \Pi \; \mathcal{N}(\Delta x, \Delta y \mid \mu, \sigma, \sigma, \rho), \text{其中} \sum_{j=1}^{M} \Pi = 1 \quad (3)$$

翻译为简体中文为：$\Pi = 1$ (3

$\mathcal{N}(x, y \mid \mu, \mu, \sigma, \sigma, \rho)$ 是双变量正态分布的概率分布函数。每个双变量正态分布由五个参数组成：(μ, μ, σ, σ, ρ)，其中 μ 和 μ 是均值，σ 和 σ 是标准差，ρ 是每个双变量正态分布的相关参数。另外，长度为 M 的向量 Π 也是一个分类分布，表示高斯混合模型的混合权重。因此，输出向量 y 的大小为 5M + M + 3，其中包括生成 (q, q, q) 所需的 3 个 logits 。

使用RNN的前向操作生成的下一个隐藏状态，通过全连接层映射到输出向量y

$$x = [S; z], [h; c] = \text{前向传播}(x, [h; c]), y = W_y h + b_y, y \in \mathbb{R}^{M+3} \quad (4)$$

向量y被分解为下一个数据点的概率分布参数

$$[(\hat{\Pi}_\mu \hat{\mu}_\sigma \hat{\sigma}_\rho) \ldots (\hat{\Pi}_\mu \hat{\mu}_\sigma \hat{\sigma}_\rho)(\hat{q}_q \hat{q}_q)] = y \quad (5)$$

As in [6], we apply $\exp$ and $\tanh$ operations to ensure the standard deviation values are non-negative, and that the correlation value is between -1 and 1:

$$\sigma_x = \exp(\hat{\sigma}_x),\ \sigma_y = \exp(\hat{\sigma}_y),\ \rho_{xy} = \tanh(\hat{\rho}_{xy}) \tag{6}$$

The probabilities for the categorical distributions are calculated using the outputs as logit values:

$$q_k = \frac{\exp(\hat{q}_k)}{\sum_{j=1}^{3}\exp(\hat{q}_j)}, k \in \{1,\ 2,\ 3\},\ \Pi_k = \frac{\exp(\hat{\Pi}_k)}{\sum_{j=1}^{M}\exp(\hat{\Pi}_j)}, k \in \{1,\ \ldots,\ M\} \tag{7}$$

A key challenge is to train our model to know when to stop drawing. Because the probabilities of the three pen stroke events are highly unbalanced, the model becomes more difficult to train. The probability of a p_1 event is much higher than p_2, and the p_3 event will only happen once per drawing. The approach developed in [7] and later followed by [26] was to use different weightings for each pen event when calculating the losses, such as a hand-tuned weighting of $(1, 10, 100)$. We find this approach to be inelegant and inadequate for our dataset of diverse image classes.

We develop a simpler, more robust approach that works well for a broad class of sketch drawing data. In our approach, all sequences are generated to a length of $N_{\max}$ where $N_{\max}$ is the length of the longest sketch in our training dataset. In principle $N_{\max}$ can be considered a hyper parameter. As the length of S is usually shorter than $N_{\max}$, we set S_i to be $(0, 0, 0, 0, 1)$ for $i > N_s$. We discuss the training in detail in the next section.

After training, we can sample sketches from our model. During the sampling process, we generate the parameters for both GMM and categorical distributions at each time step, and sample an outcome S'_i for that time step. Unlike the training process, we feed the sampled outcome S'_i as input for the next time step. We continue to sample until $p_3 = 1$, or when we have reached $i = N_{\max}$. Like the encoder, the sampled output is not deterministic, but a random sequence, conditioned on the input latent vector z. We can control the level of randomness we would like our samples to have during the sampling process by introducing a temperature parameter τ:

$$\hat{q}_k \to \frac{\hat{q}_k}{\tau},\ \hat{\Pi}_k \to \frac{\hat{\Pi}_k}{\tau},\ \sigma_x^2 \to \sigma_x^2\tau,\ \sigma_y^2 \to \sigma_y^2\tau \tag{8}$$

We can scale the softmax parameters of the categorial distribution and also the σ parameters of the bivariate normal distribution by a temperature parameter τ, to control the level of randomness in our samples. τ is typically set between 0 and 1. In the limiting case as $\tau \to 0$, our model becomes deterministic and samples will consist of the most likely point in the probability density function. Figure 3 illustrates of effect of sampling sketches with various temperature parameters.

3.3 Unconditional Generation

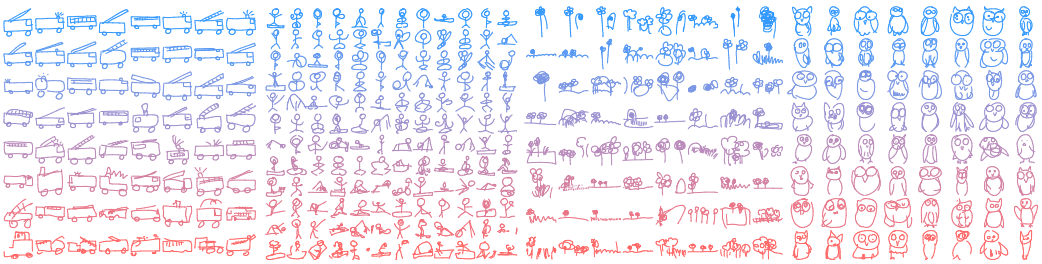

Figure 3: Unconditional generation of firetrucks, yoga poses, gardens and owls with varying τ.

As a special case, we can also train our model to generate sketches unconditionally, where we only train the decoder RNN module, without any input or latent vectors. By removing the encoder, the decoder RNN as a standalone model is an autoregressive model without latent variables. In this use case, the initial hidden states and cell states of the decoder RNN are initialized to zero. The inputs x_i of the decoder RNN at each time step is only S_{i-1} or S'_{i-1}, as we do not need to concatenate a latent vector z. In Figure 3, we sample various sketch images generated unconditionally by varying the temperature parameter from $\tau = 0.2$ at the top in blue, to $\tau = 0.9$ at the bottom in red.

与[6]中一样，我们应用exp 和tanh 操作来确保标准差值为非负，并且相关系数值在- 1和1之间

$$\sigma = \exp(\hat{\sigma}), \sigma = \exp(\hat{\sigma}), \rho = \tanh(\hat{\rho}) \tag{6}$$

使用输出作为逻辑值来计算分类分布的概率

$$q = \frac{\exp(\hat{q})}{\sum_{j=1}^{k} \exp(\hat{q})}, k = \{1, 2, 3\}, \Pi = \frac{\exp(\hat{\Pi})}{\sum_{j=1}^{M} \exp(\hat{\Pi})}, k \in \{1, \ldots, M\} \tag{7}$$

翻译为简体中文为：e的q次方

一个关键的挑战是训练我们的模型知道何时停止绘制。由于三种笔画事件的概率高度不平衡，模型变得更难训练。pevent 的概率比p高得多，并且每次绘制只会发生一次pevent 。在计算损失时，[7]中开发的方法，后来被[26]采用，是使用不同的权重来衡量每个笔事件，例如手动调整的权重(1, 10, 100)。我们发现这种方法对于我们的多样化图像类别数据集来说不够优雅和不足。

我们开发了一种更简单、更稳健的方法，适用于广泛的素描绘图数据。在我们的方法中，所有序列都生成到长度为N的长度，其中N是我们训练数据集中最长素描的长度。原则上，N可以被视为一个超参数。由于S的长度通常比N短，我们设置S为(0, 0, 0, 0, 1)，对于i > N。我们将在下一节详细讨论训练。

训练后，我们可以从我们的模型中抽样素描。在抽样过程中，我们在每个时间步生成GMM和分类分布的参数，并对该时间步的结果S进行抽样。与训练过程不同，我们将抽样结果S作为下一个时间步的输入。我们继续抽样直到p=1，或者当我们达到i=N时。与编码器一样，抽样输出不是确定性的，而是一个随机序列，以输入潜在向量z为条件。我们可以通过引入一个温度参数τ来控制在抽样过程中我们希望样本具有的随机程度。

$$\hat{q} \to q, \hat{\Pi} \to \frac{\hat{\Pi}}{\tau}, \sigma \to \sigma\tau, \sigma, \to \sigma\tau \tag{8}$$

我们可以通过温度参数τ来调整分类分布的softmax 参数和双变量正态分布的σ参数，以控制样本中的随机程度。τ通常设置在0和1之间。在τ趋近于0的极限情况下，我们的模型变得确定性，样本将由概率密度函数中最可能的点组成。

$$\to 0$$

图3展示了使用不同温度参数进行采样草图的效果。

3.3 无条件生成

图3： 使用不同的τ无条件生成消防车、瑜伽姿势、花园和猫头鹰。

作为一个特例，我们也可以训练我们的模型无条件地生成草图，即只训练解码器RNN模块，没有任何输入或潜在向量。通过移除编码器，解码器RNN作为一个独立的模型是一个没有潜在变量的自回归模型。在这种用例中，解码器RNN的初始隐藏状态和细胞状态被初始化为零。解码器RNN在每个时间步的输入x只是S或S，因为我们不需要连接一个潜在向量z。在图3中，我们通过改变温度参数从τ = 0.2（蓝色）到τ = 0.9 （红色）来无条件地采样生成各种草图图像。

3.4 Training

Our training procedure follows the approach of the Variational Autoencoder [15], where the loss function is the sum of two terms: the Reconstruction Loss, L_R, and the Kullback-Leibler Divergence Loss, L_{KL}. We train our model to optimize this two-part loss function. The Reconstruction loss term, described in Equation 9, maximizes the log-likehood of the generated probability distribution to explain the training data S. We can calculate this reconstruction loss, L_R, using the generated parameters of the pdf and the training data S. L_R is 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 (p_1, p_2, p_3), L_p:

$$L_s = -\frac{1}{N_{\max}} \sum_{i=1}^{N_s} \log \Big(\sum_{j=1}^{M} \Pi_{j,i}\, \mathcal{N}(\Delta x_i, \Delta y_i \mid \mu_{x,j,i}, \mu_{y,j,i}, \sigma_{x,j,i}, \sigma_{y,j,i}, \rho_{xy,j,i}) \Big)$$
$$L_p = -\frac{1}{N_{\max}} \sum_{i=1}^{N_{\max}} \sum_{k=1}^{3} p_{k,i} \log(q_{k,i}),\ L_R = L_s + L_p \tag{9}$$

Note that we discard the pdf parameters modelling the $(\Delta x, \Delta y)$ points beyond N_s when calculating L_s, while L_p is calculated using all of the pdf parameters modelling the (p_1, p_2, p_3) points until $N_{\max}$. Both terms are normalized by the total sequence length $N_{\max}$. We found this methodology of loss calculation to be more robust and allows the model to easily learn when it should stop drawing, unlike the earlier mentioned method of assigning importance weightings to p_1, p_2, and p_3.

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. We use the result in [15], and calculate the KL loss term, L_{KL}, normalized by number of dimensions N_z of the latent vector:

$$L_{KL} = -\frac{1}{2N_z}\Big(1 + \hat{\sigma} - \mu^2 - \exp(\hat{\sigma})\Big) \tag{10}$$

The loss function in Equation 11 is a weighted sum of both the L_R and L_{KL} loss terms:

$$Loss = L_R + w_{KL} L_{KL} \tag{11}$$

There is a tradeoff between optimizing for one term over the other. As $w_{KL} \to 0$, our model approaches a pure autoencoder, sacrificing the ability to enforce a prior over our latent space while obtaining better reconstruction loss metrics. Note that for unconditional generation, where our model is the standalone decoder, there will be no L_{KL} term as we only optimize for L_R.

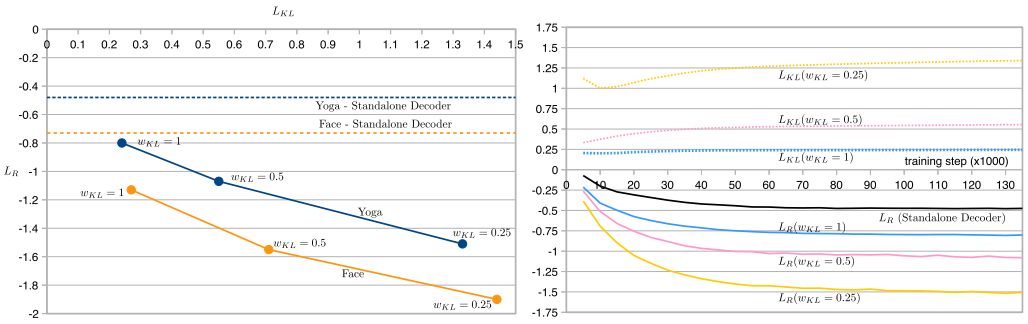

Figure 4: Tradeoff between L_R and L_{KL}, for two models trained on single class datasets (left). Validation Loss Graph for models trained on the Yoga dataset using various w_{KL}. (right)

Figure 4 illustrates the tradeoff between different settings of w_{KL} and the resulting L_R and L_{KL} metrics on the test set, along with the L_R metric on a standalone decoder RNN for comparison. As the unconditional model does not receive any prior information about the entire sketch it needs to generate, the L_R metric for the standalone decoder model serves as an upper bound for various conditional models using a latent vector.

4 Experiments

We conduct several experiments with `sketch-rnn` for both conditional and unconditional vector image generation. We train `sketch-rnn` on various `QuickDraw` classes using various settings for

3.4 训练

我们的训练过程遵循变分自动编码器[15]的方法，其中损失函数是重构损失L和Kullback - Leibler 散度损失L的总和。我们训练模型来优化这个由两部分组成的损失函数。重构损失项，描述在方程9中，最大化生成的概率分布解释训练数据S的对数似然。我们可以使用生成的概率密度函数的参数和训练数据S来计算这个重构损失L。L由偏移项的对数损失(Δx, Δy) L和笔状态项的对数损失(p, p, p) L的和组成。

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\sum_{j=1}^{M}\Pi_{j}\mathcal{N}(\Delta x, \Delta y \mid \mu, \sigma, \sigma, \rho)\right) \tag{9}$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{3}p\log(q), \quad L = L + L$$

请注意，在计算L时，我们会丢弃模拟(Δx, Δy)点超过N的pdf 参数，而L是使用模拟(p, p, p)点直到N的所有pdf 参数计算的。这两个术语都通过总序列长度N进行归一化。我们发现这种损失计算方法更加稳健，并且使模型能够轻松学习何时停止绘制，而不像之前提到的给予p、p和p重要性权重的方法。

Kullback - Leibler （KL）散度损失项衡量了我们的潜在向量z的分布与零均值和单位方差的独立同分布高斯向量之间的差异。优化这个损失项可以使我们最小化这种差异。我们使用[15]中的结果，并计算潜在向量的维度数N的KL损失项L的归一化值。

$$L = -\frac{1}{2N}\left(1 + \hat{}\sigma \text{ 翻译为简体中文为} \quad \text{简体中文翻译为：e的σ次方} \quad (10)\right) - \mu -$$

方程11中的损失函数是Land Lloss 项的加权和

$$\text{损失} = L + wL \tag{11}$$

在优化一个术语而不是另一个术语之间存在权衡。

$$\rightarrow 0$$

我们的模型接近于一个纯自编码器，牺牲了对潜在空间施加先验的能力，同时获得更好的重构损失指标。请注意，在无条件生成的情况下，我们的模型是独立的解码器，因此不会有Lterm ，我们只优化L。

图4：在单类数据集上训练的两个模型之间的土地L的权衡（左）。
在使用不同w训练的Yoga 数据集模型的验证损失图表。（右侧）

图4展示了不同wand 设置和测试集上得到的Land L指标之间的权衡，以及用于比较的独立解码器RNN的L指标。由于无条件模型没有接收到关于需要生成的整个草图的任何先验信息，独立解码器模型的L指标作为使用潜在向量的各种条件模型的上限。

4 实验

我们对sketch - rnn 进行了几个实验，包括有条件和无条件的矢量图像生成。我们使用不同的设置在各种QuickDraw 类别上训练sketch - rnn 。

w_{KL} and record the breakdown of losses. To experiment with a diverse set of classes with varying complexities, we select the cat, pig, face, firetruck, garden, owl, mosquito and yoga class. We also experiment on multi-class datasets by concatenating different classes together to form (cat, pig) and (crab, face, pig, rabbit). The results for test set evaluation on various datasets are displayed in Table 1.

The `sketch-rnn` model treats the RNN cell as an abstract component. In our experiments, we use Long Short-Term Memory (LSTM) [9] as the encoder RNN. For the decoder RNN, we use HyperLSTM, as this type of RNN cell excels at sequence generation tasks [8]. The ability for HyperLSTM to spontaneously augment its own weights enables it to adapt to many different regimes in a large diverse dataset. For model configuration details, please see the Supplementary Material.

Dataset	$w_{KL} = 1.00$		$w_{KL} = 0.50$		$w_{KL} = 0.25$		Decoder Only
	L_R	L_{KL}	L_R	L_{KL}	L_R	L_{KL}	L_R
cat	-0.98	0.29	-1.33	0.70	-1.46	1.01	-0.57
pig	-1.14	0.22	-1.37	0.49	-1.52	0.80	-0.82
cat, pig	-1.02	0.22	-1.24	0.49	-1.50	0.98	-0.75
crab, face, pig, rabbit	-0.91	0.22	-1.04	0.40	-1.47	1.17	-0.67
face	-1.13	0.27	-1.55	0.71	-1.90	1.44	-0.73
firetruck	-1.24	0.22	-1.26	0.24	-1.78	1.10	-0.90
garden	-0.79	0.20	-0.81	0.25	-0.99	0.54	-0.62
owl	-0.93	0.20	-1.03	0.34	-1.29	0.77	-0.66
mosquito	-0.67	0.30	-1.02	0.66	-1.41	1.54	-0.34
yoga	-0.80	0.24	-1.07	0.55	-1.51	1.33	-0.48

Table 1: Loss figures (L_R and L_{KL}) for various w_{KL} settings.

The relative loss numbers are consistent with our expectations. We see that the reconstruction loss term L_R decreases as we relax the w_{KL} parameter controlling the weight for the KL loss term, and meanwhile the KL loss term L_R increases as a result. The L_R for the conditional model is strictly less than the unconditional, standalone decoder model. In Figure 4 (right), we plot validation-set loss graphs for on the yoga class for models with various w_{KL} settings. As L_R decreases, the L_{KL} term tends to increase due to the tradeoff between L_R and L_{KL}.

4.1 Conditional Reconstruction

We qualitatively assess the reconstructed sketch S' given an input sketch S. In Figure 5 (left), we sample several reconstructions at various levels of temperature τ using a model trained on the single cat class, starting at 0.01 on the left and linearly increasing to 1.0 on the right. The reconstructed cat sketches have similar properties as the input image, and occasionally add or remove details such as a whisker, a mouth, a nose, or the orientation of the tail.

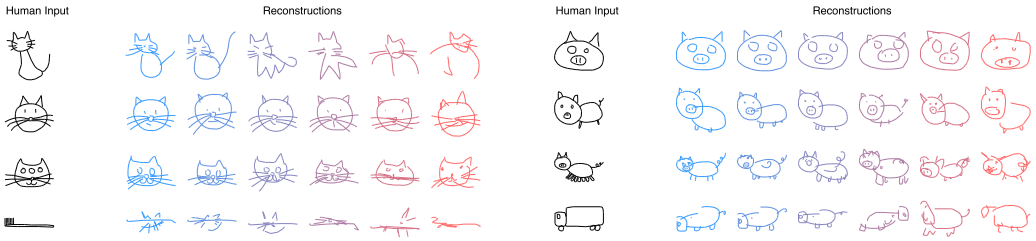

Figure 5: Conditional generation of cats (left) and pigs (right).

When presented with a non-standard image of a cat, such as a cat's face with three eyes, the reconstructed cat only has two eyes. If we input a sketch from another image class, such a toothbrush, the model seemingly generate sketches with similar orientation and properties as the toothbrush input image, but with some cat-like features such as cat ears, whiskers or feet. We perform a similar experiment with a model trained on the pig class, as shown in Figure 5 (right).

我们使用不同的设置对各种QuickDraw 类别进行sketch - rnn 训练，并记录损失的细分情况。为了尝试不同复杂度的多样化类别，我们选择了猫、猪、脸、消防车、花园、猫头鹰、蚊子和瑜伽类别。我们还通过将不同类别连接在一起形成（猫、猪）和（螃蟹、脸、猪、兔子）来对多类别数据集进行实验。各个数据集上的测试集评估结果显示在表1中。

sketch - rnn 模型将RNN单元视为一个抽象组件。在我们的实验中，我们使用长短期记忆（LSTM）[9]作为编码器RNN。对于解码器RNN，我们使用HyperLSTM ，因为这种类型的RNN单元在序列生成任务中表现出色[8]。HyperLSTM 能够自发地增强自己的权重，使其能够适应大量多样的数据集中的许多不同情况。有关模型配置详细信息，请参见补充材料。

数据集	w = 1.00		w = 0.50		w = 0.25		仅解码器			
	L	LL	L	LL	L	LL	L	LL		
猫	- 0.98	0.29	- 1.33	0.70	- 1.46	1.01	- 0.57	猪	- 1.14	0.22
- 1.37	0.49	- 1.52	0.80	- 0.82	猫，猪	- 1.02	0.22	- 1.24	0.49	
- 1.50	0.98	- 0.75	螃蟹，脸，猪，兔子	- 0.91	0.22	- 1.04	0.40	- 1.47	1.17	
- 0.67	脸	- 1.13	0.27	- 1.55	0.71	- 1.90	1.44	- 0.73	消防车	
- 1.24	0.22	- 1.26	0.24	- 1.78	1.10	- 0.90	花园	- 0.79	0.20	
- 0.81	0.25	- 0.99	0.54	- 0.62	猫头鹰	- 0.93	0.20	- 1.03	0.34	
- 1.29	0.77	- 0.66	蚊子	- 0.67	0.30	- 1.02	0.66	- 1.41	1.54	
- 0.34	瑜伽	- 0.80	0.24	- 1.07	0.55	- 1.51	1.33	- 0.48		

表1：不同w设置下的损失数值（土地L）。

相对损失数字与我们的预期一致。我们看到，当我们放松控制KL损失项权重的w参数时，重建损失项L减少，同时KL损失项L增加。条件模型的L严格小于无条件的独立解码器模型。在图4（右侧）中，我们绘制了具有不同w设置的瑜伽类模型的验证集损失图。随着L的减少，L项由于L和L之间的权衡而趋于增加。

4.1 条件重建

我们对给定的输入草图S进行定性评估重建的草图S。在图5（左）中，我们使用在单一猫类别上训练的模型，在不同温度τ的各个级别上对重建进行采样，从左侧的0.01 线性增加到右侧的1.0。重建的猫草图具有与输入图像类似的属性，并且偶尔会添加或删除细节，如胡须、嘴巴、鼻子或尾巴的方向。

图5：猫的条件生成（左）和猪的条件生成（右）。

当给出一张非标准的猫的图片，比如一只有三只眼睛的猫脸，重建后的猫只有两只眼睛。如果我们输入另一类图片的草图，比如牙刷，模型似乎会生成具有与牙刷输入图片相似方向和特性的草图，但带有一些猫的特征，比如猫耳朵、胡须或脚。我们对一个训练在猪类上的模型进行了类似的实验，如图5（右）所示。

4.2 Latent Space Interpolation

By interpolating between latent vectors, we can visualize how one image morphs into another image by visualizing the reconstructions of the interpolations. As we enforce a Gaussian prior on the latent space, we expect fewer *gaps* in the space between two encoded latent vectors. We expect a model trained using a higher w_{KL} setting to produce images that are closer to the data manifold given a spherically interpolated [24] latent vector z, compared to another model trained with a lower w_{KL}.

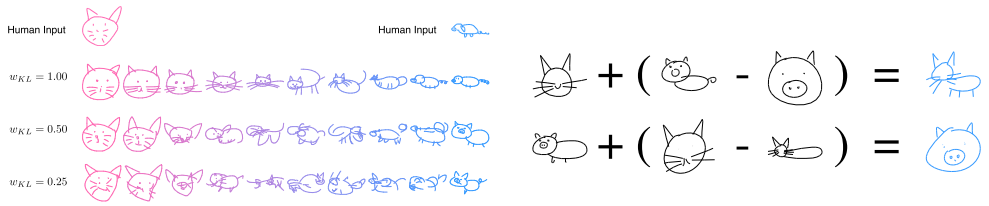

Figure 6: Latent space interpolation between cat and pig using with various w_{KL} settings (left). Sketch Drawing Analogies (right).

To demonstrate this, we train several models using various w_{KL}, on a dataset consisting of both cat and pigs, and we encode two distinct images from the test set - a cat face and a full pig. Figure 6 (left) shows the reconstructed images from the interpolated latent vectors between the two original images. As expected, models trained with higher w_{KL} produce more coherent interpolated images.

4.3 Sketch Drawing Analogies

The interpolation example in Figure 6 (left) suggests that the latent vector z encode conceptual features of a sketch. Can we use these features to augment other sketches without such features – for example, adding a body to a cat's head? Indeed, we find that sketch drawing analogies are possible for models trained with low L_{KL} numbers. Given the smoothness of the latent space, where any interpolated vector between two latent vectors results in a coherent sketch, we can perform vector arithmetic on the latent vectors encoded from different sketches and explore how the model organizes the latent space to represent different concepts in the manifold of generated sketches.

For example, as shown in Figure 6 (right), we can subtract the latent vector of an encoded pig head from the latent vector of a full pig, to arrive at a vector that represents a body. Adding this difference to the latent vector of a cat head results in a full cat (i.e. cat head + body = full cat). We repeat the experiment to remove the body of a full pig. These drawing analogies allow us to explore how the model organizes its latent space to represent different concepts in the manifold of generated sketches.

4.4 Predicting Different Endings of Incomplete Sketches

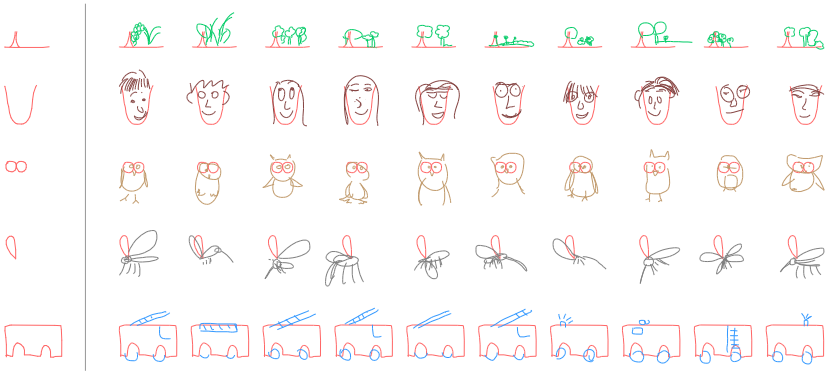

Figure 7: `sketch-rnn` predicting possible endings of various incomplete sketches (the red lines).

We can use `sketch-rnn` to finish an incomplete sketch. By using the decoder RNN as a standalone model, we can generate a sketch that is conditioned on the previous points. We use the decoder RNN to first *encode* an incomplete sketch into a hidden state h. Afterwards, we generate the remaining points of the sketch using h as the initial hidden state. We show results in Figure 7 using decoder-only models trained on individual classes, and sample completions by setting $\tau = 0.8$.

4.2 潜空间插值

通过在潜在向量之间进行插值，我们可以通过可视化插值的重构来展示一幅图像如何变形为另一幅图像。由于我们在潜在空间中施加了高斯先验，我们预期在两个编码的潜在向量之间的空间中会有较少的间隙。我们预期使用更高w值训练的模型在球形插值的潜在向量z上产生的图像会更接近数据流形，而相比之下，使用较低w值训练的另一个模型则不会。

图6：使用不同的w设置在猫和猪之间进行潜空间插值（左）。
素描绘画类比（右）。

为了证明这一点，我们使用不同的w训练了几个模型，数据集包含猫和猪的图像，并对测试集中的两个不同图像进行编码 - 一张猫脸和一只完整的猪。图6（左）显示了两个原始图像之间插值潜变量的重建图像。如预期的那样，使用较高w训练的模型产生了更连贯的插值图像。

4.3 素描绘画类比

图6（左）中的插值示例表明，潜在向量z编码了草图的概念特征。我们能否利用这些特征来增强其他没有这些特征的草图，例如给猫的头部添加身体？事实上，我们发现对于使用低L数字训练的模型，草图绘制类比是可能的。鉴于潜在空间的平滑性，在两个潜在向量之间进行插值会得到一个连贯的草图，我们可以对从不同草图编码的潜在向量进行向量运算，并探索模型如何组织潜在空间以在生成的草图流形中表示不同的概念。

例如，如图6所示（右图），我们可以从一个完整的猪的潜在向量中减去编码的猪头的潜在向量，得到代表身体的向量。将这个差异添加到猫头的潜在向量中，就得到一个完整的猫（即猫头+身体=完整的猫）。我们重复实验以去除完整猪的身体。这些绘画类比使我们能够探索模型如何组织其潜在空间以在生成的草图流形中表示不同的概念。

4.4 预测不完整草图的不同结局

图7：sketch - rnn 预测各种不完整草图的可能结尾（红线）。

我们可以使用sketch - rnn 来完成一个不完整的草图。通过将解码器RNN作为一个独立的模型使用，我们可以生成一个以前面的点为条件的草图。我们使用解码器RNN将一个不完整的草图编码成一个隐藏状态h。然后，我们使用h作为初始隐藏状态生成草图的剩余点。我们在图7中展示了使用仅解码器模型在各个类别上训练的结果，并通过设置$\tau = 0.8$ 来进行样本完成。

5 Applications and Future Work

We believe `sketch-rnn` will enable many creative applications. Even the decoder-only model trained on various classes can assist the creative process of an artist by suggesting many possible ways of finishing a sketch, helping artists expand their imagination. In the conditional model, exploring the latent space between different objects can potentially enable artists to find interesting intersections and relationships between different drawings. Even in the simplest use, pattern designers can apply `sketch-rnn` to generate a large number of similar, but unique designs for textile or wallpaper prints.

As we saw earlier in Section 4.1, a model trained to draw pigs can be made to draw pig-like trucks if given an input sketch of a truck. We can extend this result to applications that might help creative designers come up with abstract designs that can resonate more with their target audience. For instance, in Figure 8 (right), we feed sketches of four different chairs into our cat-drawing model to produce four "chair-like cats". We can even interpolate between the four images to explore the latent space of chair-like cats, and select from a large grid of generated designs.

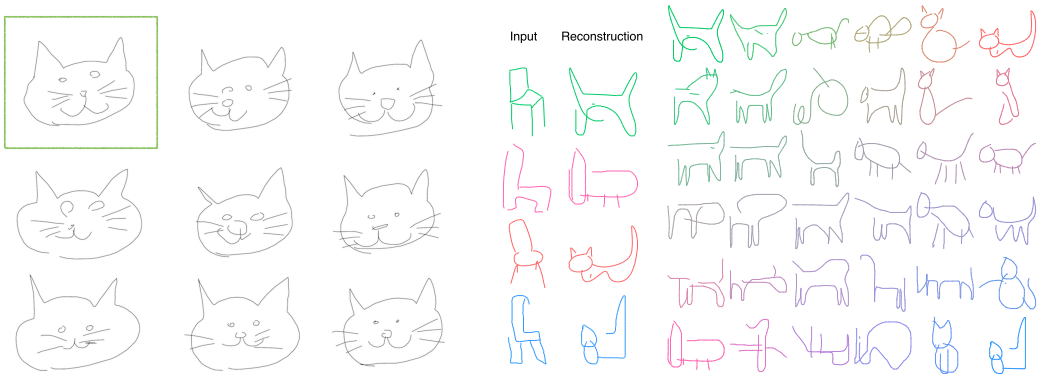

Figure 8: Generating similar, but unique sketches based on a single human sketch in the box (left). Latent space of generated cats conditioned on sketch drawings of chairs (right).

A model trained on higher quality sketches may find its way into educational applications that can help teach students how to draw. Even with the simple sketches in `QuickDraw`, the authors of this work have become much more proficient at drawing animals, insects, and various sea creatures after conducting these experiments. A related application is to encode a crude, poorly sketched drawing and generate more aesthetically looking reproductions by using a model trained with a high w_{KL} setting and sampling with a low temperature τ to produce a more coherent version of the drawing. In the future, we can also investigate augmenting the latent vector in the direction that maximizes the aesthetics of the drawing by incorporating user-rating data into the training process.

Combining hybrid variations of sequence-generation models with unsupervised, cross-domain pixel image generation models, such as Image-to-Image models [3, 13, 18], is another exciting direction that we can explore. We can already combine this model with supervised, cross-domain models such as Pix2Pix [10], to occasionally generate photo realistic cat images from generated sketches of cats. The opposite direction of converting a photograph of a cat into an unrealistic, but similar looking sketch of a cat composed of a minimal number of lines seems to be a more interesting problem.

6 Conclusion

In this work, we develop a methodology to model sketch drawings using recurrent neural networks. `sketch-rnn` is able to generate possible ways to finish an existing, but unfinished sketch drawing. Our model can also encode existing sketches into a latent vector, and generate similar looking sketches conditioned on the latent space. We demonstrate what it means to interpolate between two different sketches by interpolating between its latent space, and also show that we can manipulate attributes of a sketch by augmenting the latent space. We demonstrate the importance of enforcing a prior distribution on the latent vector for coherent vector image generation during interpolation. By making available a large dataset of sketch drawings, we hope to encourage further research and development in the area of generative vector image modelling.

5个应用和未来工作

我们相信sketch - rnn将能够实现许多创意应用。即使是仅训练在不同类别上的解码器模型，也可以通过提供许多可能的完成草图的方式来辅助艺术家的创作过程，帮助艺术家拓展他们的想象力。在条件模型中，探索不同对象之间的潜在空间可能使艺术家找到不同绘画之间有趣的交集和关系。即使在最简单的应用中，图案设计师也可以应用sketch - rnn来生成大量相似但独特的纺织品或壁纸设计。

正如我们在4.1节中所看到的，如果给出一辆卡车的草图作为输入，一个训练过绘制猪的模型也可以绘制出类似卡车的猪。我们可以将这个结果扩展到一些应用中，这些应用可能有助于创意设计师创造出更能引起目标受众共鸣的抽象设计。例如，在图8（右侧）中，我们将四张不同椅子的草图输入到我们的绘猫模型中，生成了四只“像椅子的猫”。我们甚至可以在这四张图片之间进行插值，探索椅子猫的潜在空间，并从生成的设计中选择。

图8：基于一个人的草图在方框中生成相似但独特的草图（左）。
生成的猫的潜在空间，以椅子的草图为条件（右侧）。

在高质量素描上训练的模型可能会应用于教育应用程序，帮助教学生如何绘画。即使是在QuickDraw 中的简单素描中，本研究的作者在进行这些实验后，在绘制动物、昆虫和各种海洋生物方面变得更加熟练。一个相关的应用是对粗糙、粗略的素描进行编码，并使用一个以高w设置训练的模型和低温度τ进行采样，生成更具美感的再现版本，以产生更连贯的绘画。在未来，我们还可以研究在训练过程中将用户评分数据纳入，以增强潜在向量在最大化绘画美学方面的方向。

将序列生成模型的混合变体与无监督的跨领域像素图像生成模型相结合，例如图像到图像模型[3，13, 18]，是我们可以探索的另一个令人兴奋的方向。我们已经可以将这个模型与监督的跨领域模型，如Pix2Pix [10]相结合，偶尔从生成的猫的草图中生成逼真的猫图像。将一张猫的照片转换成一个不真实但外观相似的由最少线条组成的草图的相反方向似乎是一个更有趣的问题。

6 结论

在这项工作中，我们开发了一种使用循环神经网络对草图进行建模的方法。sketch - rnn能够生成可能的方式来完成一个已经存在但未完成的草图。我们的模型还可以将现有的草图编码成潜在向量，并在潜在空间的条件下生成外观相似的草图。我们通过在潜在空间插值两个不同的草图之间展示了插值的含义，并且还展示了通过增加潜在空间来操纵草图的属性。我们展示了在插值过程中对潜在向量施加先验分布的重要性，以实现连贯的矢量图像生成。通过提供大量的草图数据集，我们希望鼓励在生成矢量图像建模领域的进一步研究和发展。

References

[1] C. M. Bishop. Mixture density networks. *Technical Report*, 1994.

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio. Generating Sentences from a Continuous Space. *CoRR*, abs/1511.06349, 2015.

[3] H. Dong, P. Neekhara, C. Wu, and Y. Guo. Unsupervised Image-to-Image Translation with Generative Adversarial Networks. *ArXiv e-prints*, Jan. 2017.

[4] M. Eitz, J. Hays, and M. Alexa. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012.

[5] I. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *ArXiv e-prints*, Dec. 2017.

[6] A. Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.

[7] D. Ha. Recurrent Net Dreams Up Fake Chinese Characters in Vector Format with TensorFlow, 2015.

[8] D. Ha, A. M. Dai, and Q. V. Le. HyperNetworks. In *ICLR*, 2017.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *ArXiv e-prints*, Nov. 2016.

[11] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. The Quick, Draw! - A.I. Experiment. *https://quickdraw.withgoogle.com/*, 2016.

[12] C. Kaae Sønderby, T. Raiko, L. Maaløe, S. Kaae Sønderby, and O. Winther. Ladder Variational Autoencoders. *ArXiv e-prints*, Feb. 2016.

[13] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *ArXiv e-prints*, Mar. 2017.

[14] D. P. Kingma, T. Salimans, and M. Welling. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016.

[15] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, Dec. 2013.

[16] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, Dec. 2015.

[17] Y. J. Lee, C. L. Zitnick, and M. F. Cohen. Shadowdraw: Real-time user guidance for freehand drawing. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 27:1–27:10, New York, NY, USA, 2011. ACM.

[18] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised Image-to-Image Translation Networks. *ArXiv e-prints*, Mar. 2017.

[19] S. Reed, A. van den Oord, N. Kalchbrenner, S. Gómez Colmenarejo, Z. Wang, D. Belov, and N. de Freitas. Parallel Multiscale Autoregressive Density Estimation. *ArXiv e-prints*, Mar. 2017.

[20] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies. *ACM Trans. Graph.*, 35(4):119:1–119:12, July 2016.

[21] M. Schuster, K. K. Paliwal, and A. General. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[22] S. Simhon and G. Dudek. Sketch interpretation and refinement using statistical models. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques*, EGSR'04, pages 23–32, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[23] P. Tresset and F. Fol Leymarie. Portrait drawing by paul the robot. *Comput. Graph.*, 37(5):348–363, Aug. 2013.

[24] T. White. Sampling Generative Networks. *ArXiv e-prints*, Sept. 2016.

[25] N. Xie, H. Hachiya, and M. Sugiyama. Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. In *ICML*. icml.cc / Omnipress, 2012.

[26] X. Zhang, F. Yin, Y. Zhang, C. Liu, and Y. Bengio. Drawing and Recognizing Chinese Characters with Recurrent Neural Network. *CoRR*, abs/1606.06539, 2016.

参考资料

C. M. Bishop . 混合密度网络。技术报告，1994 年。

S. R. Bowman ， L. Vilnis，O. Vinyals，A. M. Dai，R. Józefowicz 和S. Bengio 。从连续空间生成句子。CoRR，abs / 1511.06349，2015 年。

H. Dong ， P. Neekhara ， C. Wu和Y. Guo 。无监督的图像到图像的翻译与生成对抗网络。ArXiv电子打印，2017 年1月。

M. Eitz，J. Hays 和M. Alexa 。人类如何勾画物体？ACM Trans . Graph .（SIGGRAPH会议论文集），31（4）： 44 :1- 44 :10，2012 年。

I. Goodfellow . NIPS 2016 教程：生成对抗网络。ArXiv电子打印，2017 年12月。

[6] A. Graves . 使用循环神经网络生成序列. arXiv:1308.0850 ，2013 .

D. Ha. 2015 年，使用TensorFlow ， 循环神经网络生成矢量格式的虚假中文字符。

D. Ha, A. M. Dai, and Q. V. Le. 超网络。在ICLR，2017 年。

S. Hochreiter 和J. Schmidhuber 。长短期记忆。神经计算，1997 年。

P. Isola ，J.- Y. Zhu ， T. Zhou 和A. A. Efros 。具有条件对抗网络的图像到图像的转换。ArXiv电子打印，2016 年11月。

快速绘制！- 人工智能实验。
https ://quickdraw .withgoogle .com /, 2016 .

[12] C. Kaae Sønderby , T. Raiko , L. Maaløe , S. Kaae Sønderby , and O. Winther . 梯度变分自编码器。ArXiv电子打印版，2016 年2月。

T. Kim，M. Cha ， H. Kim，J. Lee 和J. Kim。学习使用生成对抗网络发现跨领域关系。ArXiv电子打印，2017 年3月。

D. P. Kingma ， T. Salimans 和M. Welling 。使用逆自回归流改进变分推断。CoRR ， abs / 1606.04934，2016 年。

D. P. Kingma 和M. Welling 。自动编码变分贝叶斯。ArXiv电子打印，2013 年12月。

[16] B. M. Lake , R. Salakhutdinov , and J. B. Tenenbaum . 通过概率程序归纳实现人类水平的概念学习。科学，350 (6266)：1332 - 1338，2015 年12月。

[17] Y. J. Lee , C. L. Zitnick , and M. F. Cohen . Shadowdraw ：实时用户指导的自由手绘。在ACM SIGGRAPH 2011论文集中，SIGGRAPH '11，第27 页： 1- 27： 10，美国纽约，2011年。ACM。

[18] 刘明宇，布鲁尔，J.考茨。无监督的图像到图像的翻译网络。ArXiv电子打印，2017 年3 月。

[19] S. Reed , A. van den Oord , N. Kalchbrenner , S. Gómez Colmenarejo , Z. Wang , D. Belov , and N. de Freitas .

并行多尺度自回归密度估计。ArXiv电子打印件，2017 年3 月。

P. Sangkloy ， N. Burnell ， C. Ham 和J. Hays 。《Sketchy 数据库：学习检索糟糕绘制的兔子》。ACM Trans . Graph .，35 （4）： 119：1- 119：12，2016 年7月。

[21] M. Schuster , K. K. Paliwal , and A. General . 双向循环神经网络。IEEE信号处理杂志，1997 年。

[22] S. Simhon 和G. Dudek 。使用统计模型进行草图解释和优化。在会议论文集中。

第十五届欧洲计算机图形学会议渲染技术分会议论文集，EGSR '04 ， 瑞士艾尔拉维尔，2004 年。欧洲计算机图形学协会。

[23] P. Tresset 和F. Fol Leymarie 。Paul the robot 进行的肖像画绘制。计算机图形学，37 (5)：348 - 363，8 月。2013 .

[24] T. White . 生成网络的采样。ArXiv电子打印，2016 年9 月。

[25] 谢宁，八谷浩，杉山光。艺术家代理：东方水墨画中自动笔触生成的强化学习方法。在ICML会议上。icml .cc / Omnipress ， 2012 年。

[26] 张晓，尹飞，张勇，刘超，和Bengio Y. 用循环神经网络绘制和识别汉字。CoRR ， abs / 1606.06539，2016年。

Supplementary Material for "A Neural Representation of Sketch Drawings"

1 Dataset Details

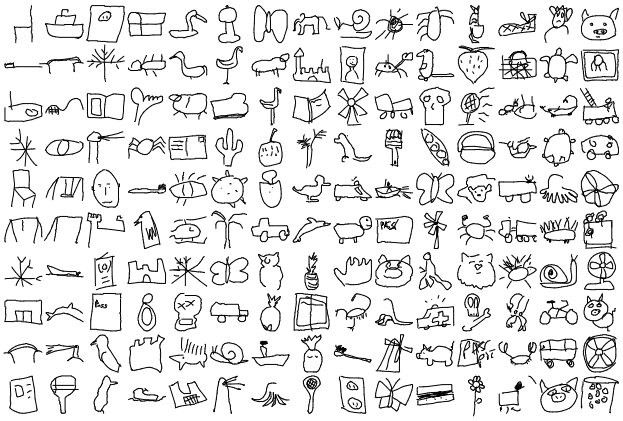

Figure 1: Example sketch drawings from `QuickDraw` dataset.

The data from *Quick, Draw!* [4] expands daily, and every so often new classes are added to the game. As such, the `QuickDraw` dataset now consists of hundreds of classes, from 75 classes initially, in Table 1. Each class consists of 70K training samples and 2.5K validation and test samples. Stroke simplification using the Ramer–Douglas–Peucker algorithm [3] with a parameter of $\epsilon = 2.0$ has been applied to simplify the lines. The data was originally recorded in pixel-dimensions, so we normalized the offsets $(\Delta x, \Delta y)$ using a single scaling factor. This scaling factor was calculated to adjust the offsets in the training set to have a standard deviation of 1. For simplicity, we do not normalize the offsets $(\Delta x, \Delta y)$ to have zero mean, since the means are already relatively small.

alarm clock	ambulance	angel	ant	barn	basket	bee
bicycle	book	bridge	bulldozer	bus	butterfly	cactus
castle	cat	chair	couch	crab	cruise ship	dolphin
duck	elephant	eye	face	fan	fire hydrant	firetruck
flamingo	flower	garden	hand	hedgehog	helicopter	kangaroo
key	lighthouse	lion	map	mermaid	octopus	owl
paintbrush	palm tree	parrot	passport	peas	penguin	pig
pineapple	postcard	power outlet	rabbit	radio	rain	rhinoceros
roller coaster	sandwich	scorpion	sea turtle	sheep	skull	snail
snowflake	speedboat	spider	strawberry	swan	swing set	tennis racquet
	the mona lisa	toothbrush	truck	whale	windmill	

Table 1: Initial 75 `QuickDraw` classes used for this work.

Figure 2 below shows a training example, before normalization of $(\Delta x, \Delta y)$ columns.

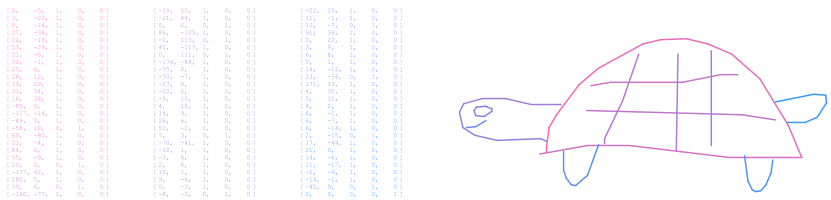

Figure 2: A sample sketch, as a sequence of $(\Delta x, \Delta y, p_1, p_2, p_3)$ points and in rendered form. In the rendered sketch, the line color corresponds to the sequential stroke ordering.

《素描绘画的神经表征》的补充材料

1 数据集详情

图1：来自QuickDraw 数据集的示例素描绘画。

从Quick , Draw ! [4]的数据每天都在扩充，而且游戏中不时会添加新的类别。因此，QuickDraw 数据集现在包含了数百个类别，最初有75 个类别，见表1。每个类别包含70K 个训练样本和2.5K 个验证和测试样本。使用Ramer - Douglas - Peucker 算法[3]对笔画进行了简化，参数为ε=2.0 。数据最初以像素尺寸记录，因此我们使用一个缩放因子对偏移量(Δx, Δy)进行了归一化。这个缩放因子是通过调整训练集中的偏移量使其标准差为1来计算的。为了简化起见，我们没有将偏移量(Δx, Δy)归一化为零均值，因为均值已经相对较小。

闹钟 救护车 天使 蚂蚁 谷仓 篮子 蜜蜂 自行车 书 桥 推土机 公共汽车 蝴蝶 仙人掌 城堡 猫 椅子 沙发 螃蟹 邮轮 海豚 鸭子 大象 眼睛 脸 扇子 消防栓 消防车 火烈鸟 花园 手 刺猬 直升机 袋鼠 钥匙 灯塔 狮子 地图 美人鱼 章鱼 猫头鹰 画笔 棕榈树 鹦鹉 护照 豌豆 企鹅 猪 菠萝 明信片 电源插座 兔子 收音机 雨 犀牛 过山车 三明治 蝎子 海龟 绵羊 骷髅 蜗牛 雪花 快艇 蜘蛛 草莓 天鹅 秋千 网球拍

蒙娜丽莎牙刷卡车鲸风车

表1：本研究使用的初始75 个QuickDraw 类别。

图2显示了一个训练示例，在对(Δx, Δy)列进行归一化之前。

图2：一个示例草图，以（Δx，Δy，p，p，p）点的序列形式呈现，并以渲染形式展示。在渲染的草图中，线条颜色对应着顺序的笔画顺序。

2 Training Details

As a recap from the main text, we defined the Reconstruction loss term L_R) as:

$$
\begin{aligned}
L_s &= -\frac{1}{N_{\max}} \sum_{i=1}^{N_s} \log \Big(\sum_{j=1}^{M} \Pi_{j,i} \, \mathcal{N}(\Delta x_i, \Delta y_i \mid \mu_{x,j,i}, \mu_{y,j,i}, \sigma_{x,j,i}, \sigma_{y,j,i}, \rho_{xy,j,i}) \Big) \\
L_p &= -\frac{1}{N_{\max}} \sum_{i=1}^{N_{\max}} \sum_{k=1}^{3} p_{k,i} \log(q_{k,i}) \\
L_R &= L_s + L_p
\end{aligned}
\tag{1}
$$

We also defined the KL loss term L_{KL} as:

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

The loss function in Equation 3 is a weighted sum of both the L_R and L_{KL} loss terms:

$$
Loss = L_R + w_{KL} L_{KL} \tag{3}
$$

While the loss function in Equation 3 can be used during training, we find that annealing the KL term in the loss function (Equation 4) produced better results. This modification is only used for model training, and the original loss function in Equation 3 is still used to evaluate validation and test sets, and for early stopping.

$$
\begin{aligned}
\eta_{\texttt{step}} &= 1 - (1 - \eta_{\texttt{min}}) R^{\texttt{step}} \\
Loss_{\texttt{train}} &= L_R + w_{KL} \; \eta_{\texttt{step}} \; \max(L_{KL}, KL_{\texttt{min}})
\end{aligned}
\tag{4}
$$

We find that annealing the KL loss term generally results in better losses. Annealing the L_{KL} term in the loss function directs the optimizer to first focus more on the reconstruction term in Equation 1, which is the more difficult loss term of the model to optimize for, before having to deal with optimizing for the KL loss term in Equation 2, a far simpler expression in comparison. This approach has been used in [2, 5, 7]. Our annealing term $\eta_{\texttt{step}}$ starts at $\eta_{\texttt{min}}$ (typically 0 or 0.01) at training step 0, and converges to 1 for large training steps. R is a term close to, but less than 1.

If the distribution of z is close enough to $\mathcal{N}(0, I)$, we can sample sketches from the decoder using randomly sampled z from $\mathcal{N}(0, I)$ as the input. In practice, we find that going from a larger L_{KL} value ($L_{KL} > 1.0$) to a smaller L_{KL} value of ~ 0.3 generally results in a substantial increase in the quality of sampled images using randomly sampled $z \sim \mathcal{N}(0, I)$. However, going from $L_{KL} = 0.3$ to L_{KL} values closer to zero does not lead to any further noticeable improvements. Hence we find it useful to put a floor on L_{KL} in the loss function by enforcing $\max(L_{KL}, KL_{\texttt{min}})$ in Equation 4.

The $KL_{\texttt{min}}$ term inside the $\max$ operator is typically set to a small value such as 0.10 to 0.50. This term will encourage the optimizer to put less focus on optimizing for the KL loss term L_{KL} once it is low enough, so we can obtain better metrics for the reconstruction loss term L_R. This approach is similar to the approach described in [7] as *free bits*, where they apply the $\max$ operator separately inside each dimension of the latent vector z.

3 Model Configuration

Our encoder and decoder RNNs consist of 512 and 2048 nodes respectively. In our model, we use $M = 20$ mixture components for the decoder RNN. The latent vector z has $N_z = 128$ dimensions. We apply Layer Normalization [1] to our model, and during training apply recurrent dropout [9] with a keep probability of 90%. We train the model with batch sizes of 100 samples, using Adam [6] with a learning rate of 0.0001 and gradient clipping of 1.0. All models are trained with $KL_{\texttt{min}} = 0.20$, $R = 0.99999$. During training, we perform simple data augmentation by multiplying the offset columns $(\Delta x, \Delta y)$ by two IID random factors chosen uniformly between 0.90 and 1.10. Unless mentioned otherwise, all experiments are conducted with $w_{KL} = 1.00$.

2培训详情

作为主文的回顾，我们将重建损失项L定义为：

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\prod_{j=1}^{M}\mathcal{N}(\Delta x, \Delta y) \mid \mu, \sigma, \sigma, \rho)$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{2}\left(\sum \log(q)\right) \quad (1)$$

$$L = L + L$$

我们还定义了KL损失项Las：

$$L = \frac{1}{-2N}\ 1 + \hat{}\sigma \text{ 翻译为简体中文为} \quad \text{简体中文翻译为：e的σ次方} \quad (2)$$

方程3中的损失函数是Land Lloss项的加权和 $\left(\quad\right)$

$$\text{损失} = L + wL \quad (3)$$

在训练过程中可以使用方程3中的损失函数，但我们发现在损失函数中逐渐减小KL项（方程4）可以获得更好的结果。这种修改仅用于模型训练，方程3中的原始损失函数仍然用于评估验证和测试集，并用于提前停止。

$$\text{损失} = L + w\eta\max(L, KL)^{\eta=1\ (1\ )R} \quad (4)$$

我们发现，退火KL损失项通常会导致更好的损失。在损失函数中退火L项会指导优化器首先更多地关注方程1中的重构项，这是模型中更难优化的损失项，然后再处理方程2中的KL损失项，相比之下，这是一个更简单的表达式。这种方法已经在[2, 5, 7]中使用过。我们的退火项η在训练步骤0时开始为η（通常为0或0.01），并在大的训练步骤中收敛到1。R是一个接近但小于1的项。

如果z的分布足够接近 $\mathcal{N}$（0，我），我们可以使用解码器从中抽取草图样本 $\mathcal{N}$（0，我）随机抽取输入。在实践中，我们发现从较大的L值（L> 1.0）转变为较小的值，通常会导致使用随机采样的z的样本图像质量显著提高。将L从0.3 减小到接近零的值并没有带来进一步明显的改善。因此，我们发现在损失函数中通过强制max (L, KL)来设定L的下限是有用的。 $\sim 0.$

KLterm 内的max 运算符通常设置为较小的值，如0.10 至0.50 。一旦KL损失项L足够低，这个项将鼓励优化器减少对KL损失项的优化关注，从而获得更好的重构损失项L的指标。这种方法类似于[7]中描述的自由位方法，其中他们在潜在向量z的每个维度内分别应用max 运算符。

3 模型配置

我们的编码器和解码器RNN分别由512和2048 个节点组成。在我们的模型中，我们为解码器RNN使用M = 20 个混合组件。潜在向量z具有N = 128 个维度。我们在模型中应用层归一化[1]，并在训练过程中应用保持概率为90 % 的递归丢失[9]。我们使用100 个样本的批量大小来训练模型，使用学习率为0.0001 和梯度剪裁为1.0的Adam [6]进行训练。所有模型都使用KL = 0.20 ，R = 0.99999 进行训练。在训练过程中，我们通过将偏移列（Δx，Δy）乘以两个在0.90 和1.10之间均匀选择的IID随机因子来进行简单的数据增强。除非另有说明，所有实验都以w = 1.00 进行。

4 Model Limitations

Although `sketch-rnn` can model a large variety of sketch drawings, there are several limitations in the current approach we wish to highlight. For most single-class datasets, `sketch-rnn` is capable of modelling sketches up to around 300 data points. The model becomes increasingly difficult to train beyond this length. For our dataset, we applied the Ramer–Douglas–Peucker algorithm [3] to simplify the strokes of the sketch data to less than 200 data points while still keeping most of the important visual information of each sketch.

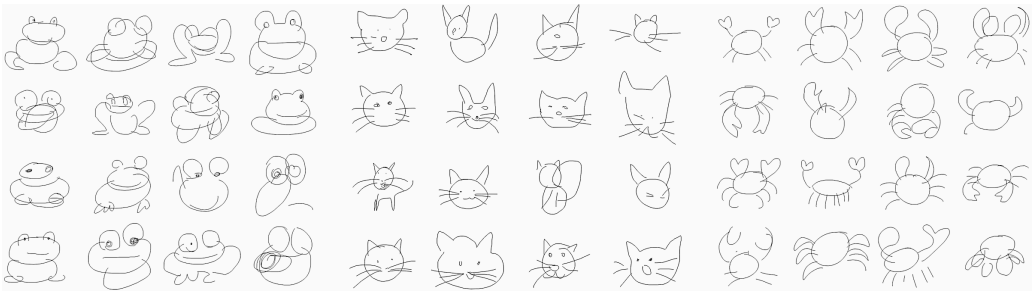

Figure 3: Unconditional generated sketches of frogs, cats, and crabs at $\tau = 0.8$.

For more complicated classes of images, such as mermaids or lobsters, the reconstruction loss metrics are not as good compared to simpler classes such as ants, faces or firetrucks. The models trained on these more challenging image classes tend to draw smoother, more circular line segments that do not resemble individual sketches, but rather resemble an averaging of many sketches in the training set. We can see some of this artifact in the frog class, in Figure 3. This smoothness may be analogous to the blurriness effect produced by a Variational Autoencoder [8] that is trained on pixel images. Depending on the use case of the model, smooth circular lines can be viewed as aesthetically pleasing and a desirable property.

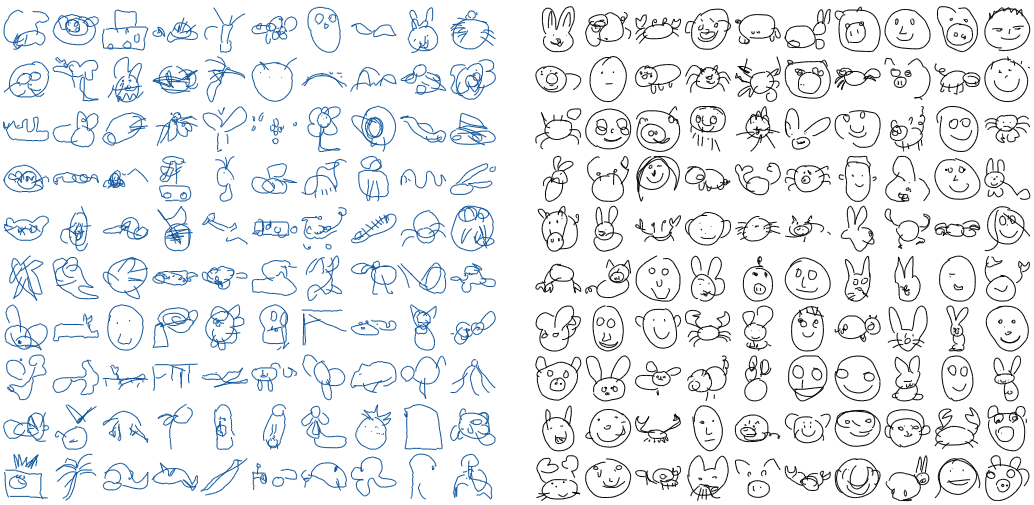

Figure 4: Unconditional generations from model trained on 75 classes (left),
From model trained on crab, face, pig and rabbit classes (right).

While both conditional and unconditional models are capable of training on datasets consisting of several classes, such as (cat, pig), and (crab, face, pig, rabbit), `sketch-rnn` is ineffective at modelling a large number of classes simultaneously. In Figure 4, we sample sketches using an unconditional model trained on 75 classes, and a model trained on 4 classes. The samples generated from the 75-class model are incoherent, with individual sketches displaying features from multiple classes. The four-class unconditional model usually generates samples of a single class, but occasionally also combines features from multiple classes. In the future, we will explore incorporating class information outside of the latent space to handle the modelling of a large number of classes simultaneously.

4个模型限制

虽然sketch - rnn可以模拟各种各样的素描图，但当前方法存在一些限制，我们希望强调一下。对于大多数单一类别的数据集，sketch - rnn能够模拟多达约300 个数据点的素描。超过这个长度，模型的训练变得越来越困难。对于我们的数据集，我们应用了Ramer - Douglas - Peucker 算法[3]来简化素描数据的笔画，使其保持在不到200 个数据点的范围内，同时仍保留大部分重要的视觉信息。

图3：$\tau = 0.8$ 时生成的青蛙、猫和螃蟹的无条件草图。

对于更复杂的图像类别，如美人鱼或龙虾，与蚂蚁、人脸或消防车等简单类别相比，重建损失指标并不好。在这些更具挑战性的图像类别上训练的模型往往会绘制出更平滑、更圆形的线段，而不是像训练集中的许多草图一样。我们可以在图3中的青蛙类别中看到这种现象。这种平滑性可能类似于在像素图像上训练的变分自动编码器[8]产生的模糊效果。根据模型的使用情况，平滑的圆形线条可以被视为具有美感和可取的特性。

图4：从训练有75个类别的模型生成的无条件生成物（左），从训练有螃蟹、脸、猪和兔子类别的模型生成的物体（右）。

虽然条件模型和无条件模型都能够在包含多个类别的数据集上进行训练，比如（猫，猪）和（螃蟹，脸，猪，兔子），但是sketch - rnn在同时建模大量类别时效果不佳。在图4中，我们使用一个在75个类别上训练的无条件模型和一个在4个类别上训练的模型来生成草图样本。从75个类别模型生成的样本是不连贯的，每个草图显示了多个类别的特征。四个类别的无条件模型通常生成单一类别的样本，但偶尔也会结合多个类别的特征。未来，我们将探索在潜在空间之外加入类别信息，以同时处理大量类别的建模问题。

5 Multi-Sketch Drawing Interpolation

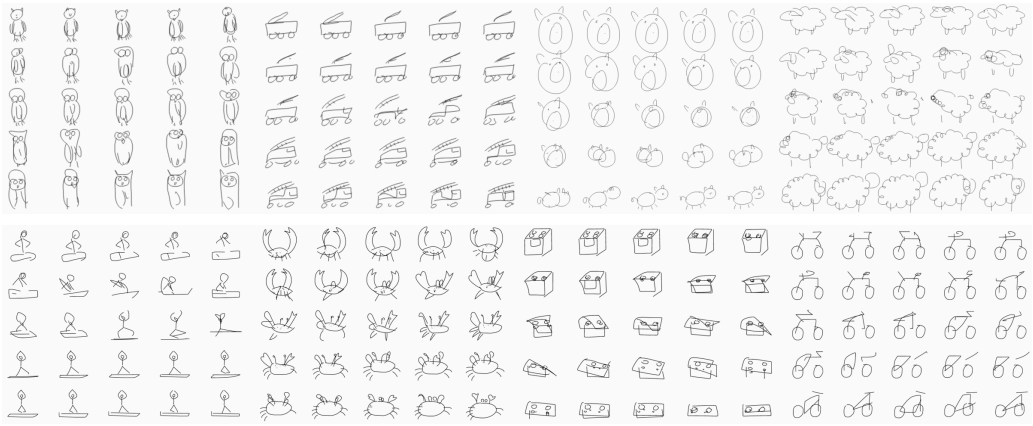

Figure 5: Example of conditional generated sketches with single class models.
Latent space interpolation from left to right, and then top to bottom.

In addition to interpolating between two sketches, like in Figure 5, we can also visualize the interpolation between four sketches in latent space to gain further insight from the model. In this section we show more examples conditionally generated with `sketch-rnn`. We take four generated images, place them on four corners of a grid, and populate the rest of the grid using the interpolation of the latent vectors at the corners. Figure 6 shows two examples of this four-way interpolation, using models trained on both (cat, pig) classes, and face class. All samples generated with $\tau = 0.1$.

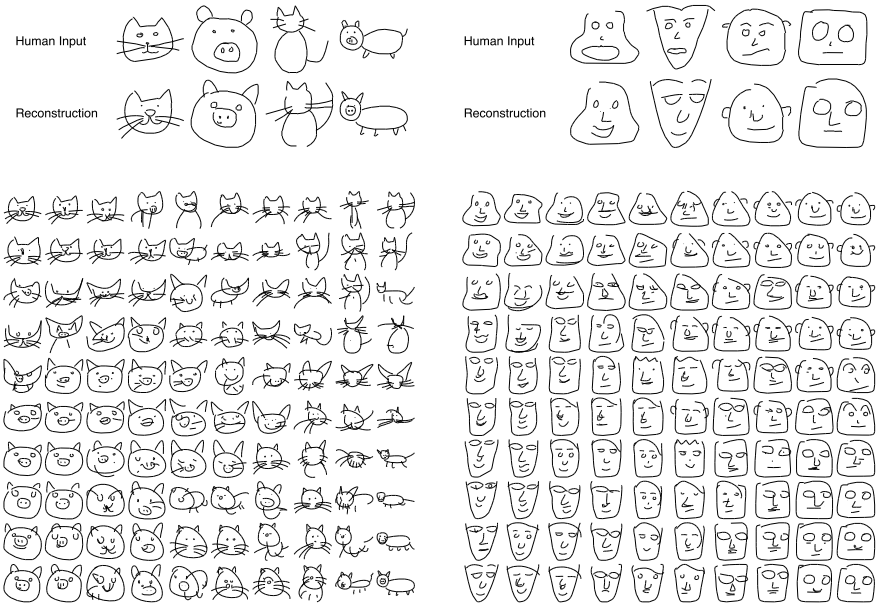

Figure 6: Example input sketches and `sketch-rnn` generated reproductions (Top),
Latent space interpolation between the four reproduced sketches (Bottom).

The left side of Figure 7 visualizes the interpolation between a full pig, a rabbit's head, a crab, and a face, using a model trained on these four classes. In certain parts of the space between a crab and a face is a rabbit's head, and we see that the ears of the rabbit becomes the crab's claws. Applying the model on the yoga class, it is interesting to see how one yoga position slowly transitions to another via a set of interpolated yoga positions generated by the model. For visual effect, we also interpolate between four distinct colors, and color each sketch using a unique interpolated color.

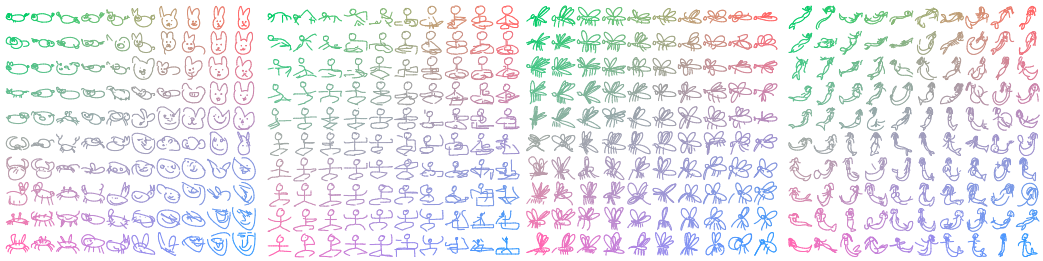

Figure 7: Interpolation of (pig, rabbit, crab and face), yoga poses, mosquitoes and mermaids. We also interpolate between four distinct colors for visual effect.

We also construct latent space interpolation examples for the mosquito class and the mermaid class, in the last two grids Figure 7. We see that the model can interpolate between concepts such as style of wings, leg counts, and orientation. In Figure 8 below, we show more interpolation examples of other classes from the dataset.

Figure 8: Latent space interpolation between four generated gardens, owls, cats, and firetrucks.

6 Which Loss Controls Image Coherency?

We would like to question the relative importance of the reconstruction loss term L_R, relative to the KL loss term L_{KL}, when our goal is to produce higher quality image reconstructions. While our reconstruction loss term L_R optimizes for the log-likelihood of the set of strokes that make up a sketch, this metric alone does not give us any guarantee that a model with a lower L_R number will produce higher quality reconstructions compared to a model with a higher L_R number.

For example, imagine a simple sketch of an face, ☺, where most of the data points of S are be used to represent the head, and only a minority of points represent facial features such as the eyes and mouth. It is possible to reconstruct the face with incoherent facial features, and yet still score a lower L_R number compared to another reconstruction with a coherent and similar face, if the edges around the incoherent face are generated more precisely.

In Figure 9, we compare the reconstructed images generated using models trained with various w_{KL} settings. In the first three examples from the left, we train our model on a dataset consisting of four image classes (crab, face, pig, rabbit). We deliberately sketch input drawings that contain features of two classes, such as a rabbit with a pig mouth and pig tail, a person with animal ears, and a rabbit with crab claws. We see that the model trained using higher w_{KL} weights, tend to generate sketches with features of a single class that look more coherent, despite having lower L_{KL} numbers. For instance, the model with $w_{KL} = 1.00$ omit pig features, animal ears, and crab claws from its reconstructions. In contrast, the model with $w_{KL} = 0.25$, with higher L_{KL}, but lower L_R numbers tries to keep both inconsistent features, while generating sketches that look less coherent.

In the last three examples in Figure 9, we repeat the experiment on models trained on single-class images, and see similar results even when we deliberately choose input samples from the test set with noisier lines.

If we look at the interpolations produced in the Latent Space Interpolation section in the main text, models with better KL loss terms also generate more meaningful reconstructions from the interpolated

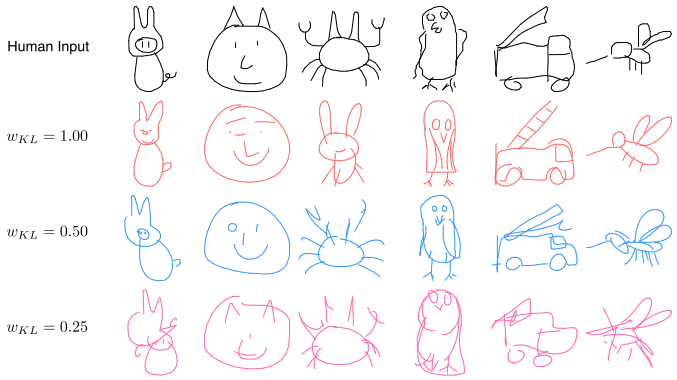

Figure 9: Reconstructions of sketch images using models with various w_{KL} settings.

space between two latent vectors. This suggests the latent vector for models with lower L_{KL} control more meaningful parts of the drawings, such as controlling whether the sketch is an animal head only or a full animal with a body, or whether to draw a cat head or a pig head. Altering such latent vectors can allow us to directly manipulate these animal features. Conversely, altering the latent codes of models with higher L_{KL} results in scattered movement of individual line segments, rather than alterations of meaningful conceptual features of the animal.

This result is consistent with incoherent reconstructions seen in Figure 9. With a lower L_{KL}, the model is likely to generate coherent images given any random z. Even with a non-standard, or noisy, input image, the model will still encode a z that produces coherent images. For models with lower L_{KL} numbers, the encoded latent vectors contain conceptual features belonging to the input image, while for models with higher L_{KL} numbers, the latent vectors merely encode information about specific line segments. This observation suggests that when using `sketch-rnn` on a new dataset, we should first try different w_{KL} settings to evaluate the tradeoff between L_R and L_{KL}, and then choose a setting for w_{KL} (and $KL_{\texttt{min}}$) that best suit our requirements.

7 Acknowledgements

We thank Ian Johnson, Jonas Jongejan, Martin Wattenberg, Mike Schuster, Thomas Deselaers, Ben Poole, Kyle Kastner, Junyoung Chung and Kyle McDonald for their help with this project. This work was done as part of the Google Brain Residency program (`g.co/brainresidency`).

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *NIPS*, 2016.

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio. Generating Sentences from a Continuous Space. *CoRR*, abs/1511.06349, 2015.

[3] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, Oct. 1973.

[4] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. The Quick, Draw! - A.I. Experiment. *https://quickdraw.withgoogle.com/*, 2016.

[5] C. Kaae Sønderby, T. Raiko, L. Maaløe, S. Kaae Sønderby, and O. Winther. Ladder Variational Autoencoders. *ArXiv e-prints*, Feb. 2016.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[7] D. P. Kingma, T. Salimans, and M. Welling. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016.

[8] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, Dec. 2013.

[9] S. Semeniuta, A. Severyn, and E. Barth. Recurrent dropout without memory loss. *arXiv:1603.05118*, 2016.