Supplementary material for
BézierSketch: A generative model for scalable
vector sketches

Ayan Das\textsuperscript{1,2}, Yongxin Yang\textsuperscript{1,2}, Timothy Hospedales\textsuperscript{1,3}, Tao Xiang\textsuperscript{1,2}, and
Yi-Zhe Song\textsuperscript{1,2}

\textsuperscript{1} SketchX, CVSSP, University of Surrey, United Kingdom
\{a.das,yongxin.yang,t.xiang,y.song\}@surrey.ac.uk
\textsuperscript{2} iFlyTek-Surrey Joint Research Centre on Artificial Intelligence
\textsuperscript{3} University of Edinburgh, United Kingdom
t.hospedales@ed.ac.uk

1 Appendix A

Property 1. Given a \((T, P)\) pair where \(T = d(P)\) for an arbitrary set of \(t\), and \(\hat{P} \sim \mathcal{N}(P, \Sigma)\), then the decoded \(\hat{T} = d(\hat{P})\) with the same set of \(t\), is distributed as \(\mathcal{N}(T, \Sigma')\), where \(\Sigma\) and \(\Sigma'\) are diagonal covariance matrices.

Proof. As \(\Sigma\) is diagonal, we can separate each dimension of \(\mathcal{N}(P, \Sigma)\) into individual Gaussians and then group \(x - y\) components of each control point with its own Gaussian with diagonal covariance \(\Sigma_i \triangleq \begin{bmatrix} \sigma_{x_i}, 0 \\ 0, \sigma_{y_i} \end{bmatrix}\)

\[
\mathcal{N}(P, \Sigma) = \prod_{i=0}^{n} \mathcal{N}(P_i, \Sigma_i)
\]

By drawing samples from the gaussians of individual control points, we get \(\hat{P} \triangleq \begin{bmatrix} \hat{P}_i \end{bmatrix}_{i=0}^{n}\) where \(\hat{P}_i \sim \mathcal{N}(P_i, \Sigma_i)\). Decoding \(\hat{P}\) by \(d(\cdot)\) gives

\[
\hat{T} = d(\hat{P}) = \sum_{i=0}^{n} B_{i,n}(t) \cdot \hat{P}_i \tag{1}
\]

Given any value of \(t = t\), the random variable \(\hat{T}\) is a weighted sum of \(n\) independent gaussian random variables with weights \([B_{i,n}(t)]_{i=0}^{n}\). Hence, \(\hat{T}\) is distributed as

\[
\hat{T} \sim \mathcal{N} \left( \sum_{i=0}^{n} B_{i,n}(t) \cdot P_i, \sum_{i=0}^{n} B_{i,n}^2(t) \cdot \Sigma_i \right) \tag{2}
\]

Now we know that \(\sum_{i=0}^{n} B_{i,n}(t) \cdot P_i \triangleq T\) and we denote \(\sum_{i=0}^{n} B_{i,n}^2(t) \cdot \Sigma_i \triangleq \Sigma'\). So,

\[
\hat{T} \sim \mathcal{N}(T, \Sigma')
\]
Sketch-RNN [2] is considered the state-of-the-art generative model for free-hand vector sketches. Sketch-RNN models the consecutive differences of 2D waypoints of a sketch along with three bits denoting “touching”, “stroke-end” and “sketch-end” state of the pen. In control point mode of BézierSketch, we adopted the same architecture and data representation as Sketch-RNN but with control points instead of waypoints. Hence, a sketch $S_{cp}$ is transformed to a list (of length $N$) of $5$-tuples $s_i \equiv (\Delta P_x, \Delta P_y, q_1, q_2, q_3)$ where $[\Delta P_x, \Delta P_y]^T \equiv \Delta P$ is the successive difference of control points and $(q_1, q_2, q_3) \equiv q$ are the three flag bits described above. As a normalization step, all sketches have been assumed to start from the origin (i.e., $[0, 0]^T$).

The core model of Sketch-RNN is a Sequence-to-Sequence Variational Autoencoder (Seq2Seq-VAE) [4] with a standard sequence encoder and an autoregressive decoder. The whole sketch sequence is fed into a Bidirectional encoder LSTM with hidden state given as

$$h_i \equiv [\overrightarrow{h_i}; \overleftarrow{h_i}] = \text{Bi-LSTM}(s_i, h_{i-1}) \quad (3)$$

and the last state $h_N$ is used as a compact representation of the sketch. $h_N$ is then used to generate the parameters of a gaussian distribution following the VAE framework [3]. A sample is then drawn from the distribution as

$$z \sim N(\mu, \sigma), \text{ where } [\mu, \sigma] = f(h_N) \in \mathbb{R}^Z \text{ and }$$

and decoded by an autoregressive decoder. An unidirectional LSTM is employed to initialize from $z$ and produce a reconstruction of the sketch sequence similar to [1]. At each time-step $j$ of the decoder, the hidden state is given as

$$g_j = \text{LSTM}(z; s_j), \text{ with } g_0 = \text{tanh}(z)$$

The decoder, at every time-step, outputs the parameters of a GMM (with $M$ mixtures) on $[\Delta P_x, \Delta P_y]^T$ and also a categorical distribution on three flag bits discussed above. Samples from these distributions are fed back as input $s_{j+1}$ at next time step

$$s'_j = (\Delta P'_j, q'_j), \text{ where }$$

$$\Delta P'_j \sim \text{GMM}(\Delta P; g_j) \text{ and } q'_j \sim \text{Cat}(q; g_j) \quad (4)$$

The network is trained with the following loss that comprises of log-likelihood of the GMM, categorical cross-entropy of the flag bits and a variational KL divergence loss

$$L = -\frac{1}{N_{\text{max}}} \left[ \sum_{j=1}^{N} \log \text{GMM}(\Delta P'_j) + \sum_{j=1}^{N_{\text{max}}} q_j \log q'_j \right]$$

$$-\frac{1}{2Z}(1 + \sigma - \mu^2 - \exp(\sigma)) \quad (5)$$
3 Appendix C

We provide visualizations (Refer to Fig. 1) of the optimization dynamics over time. We also annotate a discrete point of the stroke and its corresponding point on the Bézier curve by joining them by a connector.

![Visualization of intermediate stages of the fitting for BézierEncoder network. Each row corresponds to one sample and columns denote increasing iterations of training.](image)

**Fig. 1.** Visualization of intermediate stages of the fitting for BézierEncoder network. Each row corresponds to one sample and columns denote increasing iterations of training.

References