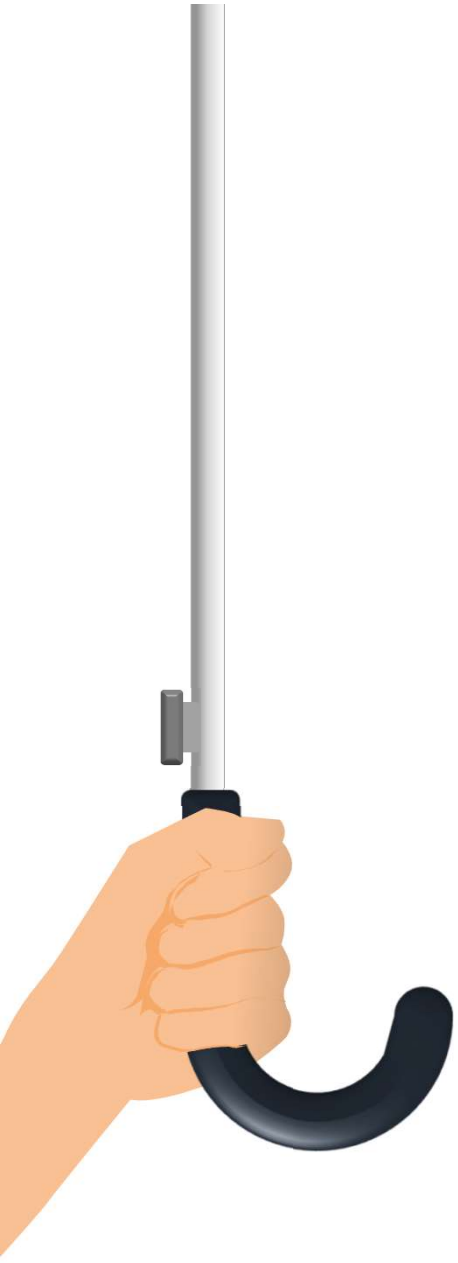


A NEURAL REPRESENTATION OF SKETCH DRAWINGS

HB

Abstract

HB

- Stroke-based drawing을 생성할 수 있는 sketch-rnn 모델을 제안한다.
- 이 모델은 사람이 그린 여러 클래스의 스케치 데이터로 훈련되었다.
- 조건부 및 무조건적 스케치 생성을 위한 프레임워크의 개요를 설명한다.
- 일관된 스케치 도면을 벡터 형식으로 생성하기 위한 새로운 강력한 train 방법을 설명한다

1. Introduction

HB

- 목표 : 인간과 비슷한 방식으로 추상적인 개념을 그리고 일반화 하도록 기계를 훈련
 - 손으로 그린 스케치 데이터 셋에 대해 모델을 train
 - 각 스케치는 펜을 제어하는 일련의 동작(이동방향, 펜을 들어올리는 시기, 그리기를 중지하는 시기)로 표현

1. Introduction

HB

- Contribution

- 일련의 선으로 구성된 벡터 이미지의 무조건 및 조건 생성을 위한 프레임워크를 개략적으로 소개

- 벡터 형식으로 스케치 생성 가능

- 벡터 이미지 데이터 세트를 제공하고 오픈 소스 프로젝트로서 모델의 구현을 공개

2. Related work

HB

- Drawing by Paul the Robot

- 기계 로봇 팔을 제어하는 알고리즘이 디지털화된 사람의 초상화를 모방하기 위한 프로그램

- 강화학습 기반으로 디지털화된 사진을 모방

- Neural Network-based approaches

- Neural Network 기반 접근방법이 개발

- 벡터 영상 생성 작업은 거의 수행되지 않음

- 초기 작품(Simhon & Dudek, 2004)은 인간 스케치의 선과 곡선을 합성하기 위해 숨겨진 마르코프 모델을 사용

- 보다 최근의 연구(Graves, 2013)는 연속적인 데이터 포인트를 생성하기 위해 혼합 밀도 네트워크(Bishop, 1994)를 활용할 수 있는 토대를 마련

- 최근 연구는 한자를 일련의 펜 스트로크 동작으로 모델링하여 벡터화된 간지 문자(Ha, 2015; Zhang 등, 2016)를 생성하려고 시도

3.1 Dataset

HB

- Quicj draw 데이터 셋(50 milion)을 구성
- 스케치를 펜 스트로크 동작 세트로 나타내었다. (이 데이터 형식에서는 도면의 초기 좌표가 원점에 위치)

- **x, y** : x 및 y 축 방향으로의 오프셋 거리

- p1,p2,p3 는 one-hot

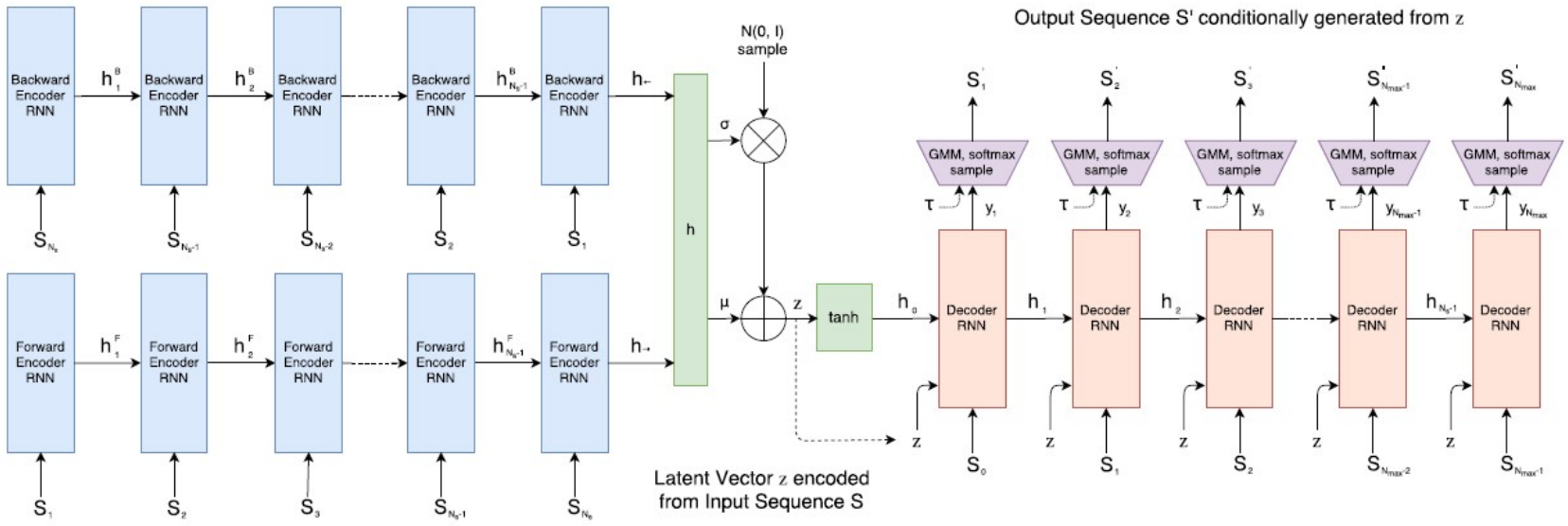
P1 : 펜이 현재 용지와 닿고 있고 다음 포인트와 현재 포인트를 연결하는 선이 그려지고 있음을 나타냄

P2 : 현재 지점 이후에 펜이 용지에서 들어 올려지고 다음에는 선이 그려지지 않음

P3 : 스케치가 끝남

3.2 Sketch-RNN

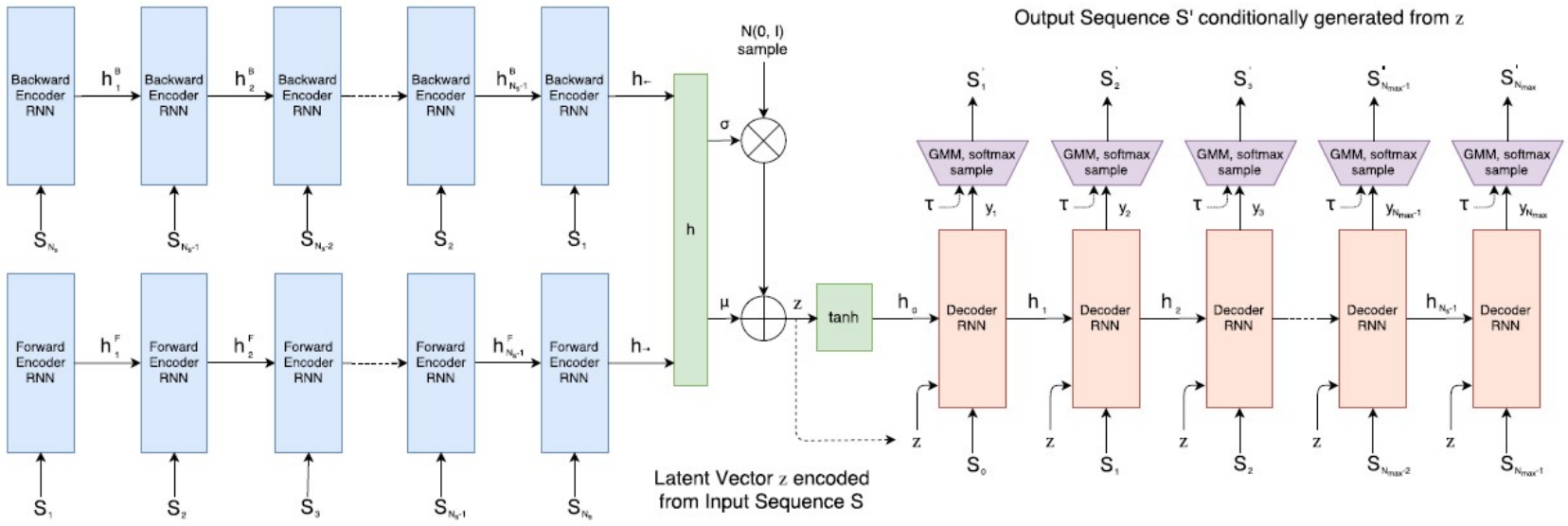
HB



- 우리의 모델은 Sequence-to-Sequence Variational Autoencoder 를 제안
- 인코더는 양방향 RNN
- 스케치를 인풋으로 받고 아웃풋으로는 레이턴트 벡터(사이즈는 N_Z)

3.2 Sketch-RNN

HB



- 우리는 스케치 시퀀스 S와 역순 시퀀스 SR을 두 인코더 RNN에 입력으로 주어 두개의 final hidden state를 얻어낸다.

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

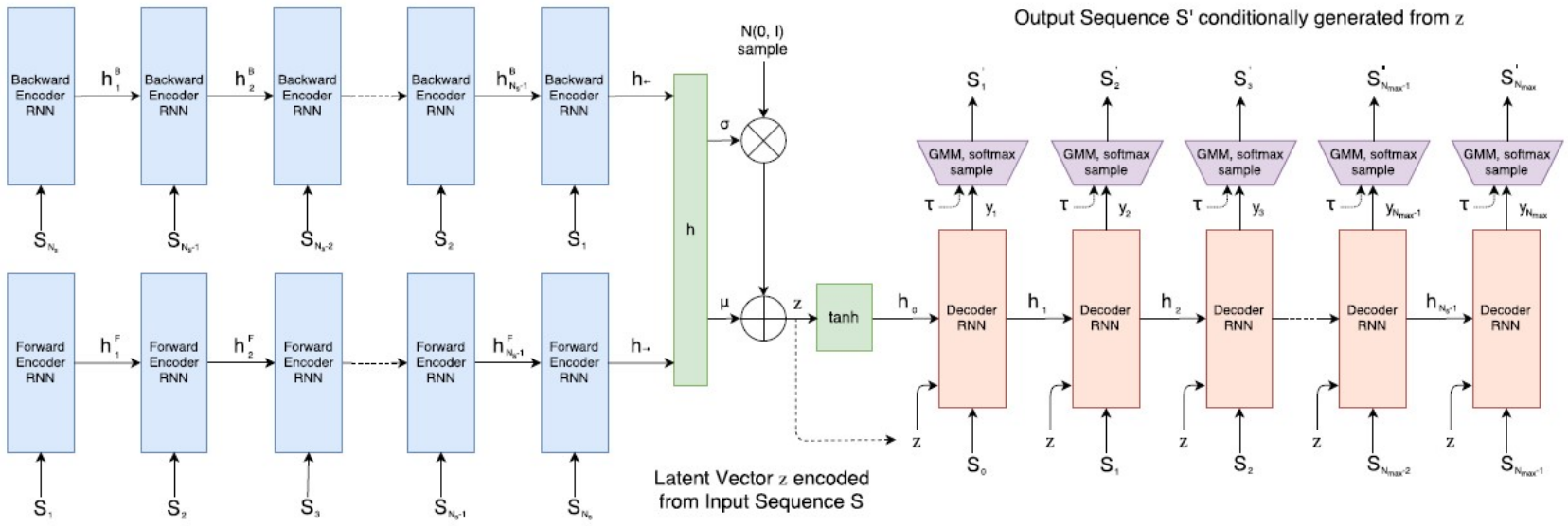
- 우리는 최종 합쳐진 히든 상태 h를 이용해 두 벡터(평균,표준편차)로 project 한다.

- 이 후 두 벡터를 이용해 IID Gaussian vector N(0,I)에 따라 random vector(z)를 생성

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

3.2 Sketch-RNN

HB



- 우리는 스케치 시퀀스 S와 역순 시퀀스 SR을 두 인코딩 RNN에 입력으로 주어 두개의 final hidden state를 얻어낸다.

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

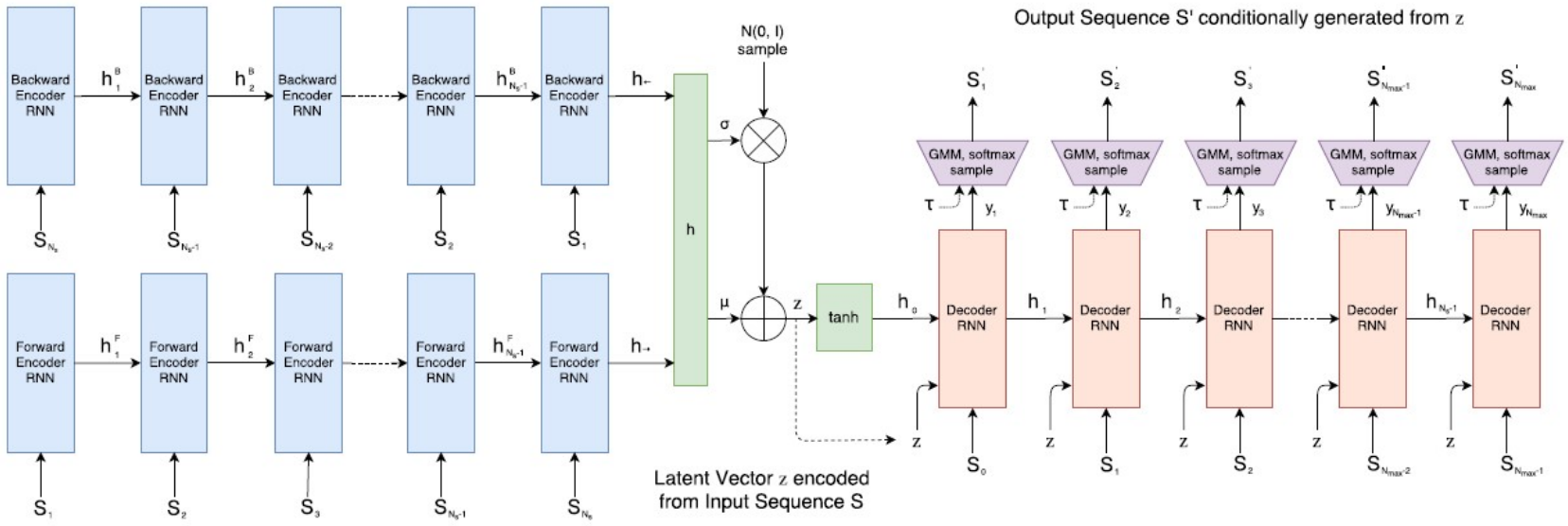
- 우리는 최종 합쳐진 히든 상태 h를 이용해 두 벡터(평균,표준편차)로 project 한다.

- 이 후 두 벡터를 이용해 IID Gaussian vector N(0,I)에 따라 random vector(z)를 생성

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

3.2 Sketch-RNN

HB



- 디코더는 주어진 latent vector z에 조건부로 출력 스케치를 샘플링 하는 auto-regressive RNN

- RNN의 초기 hidden state(h0) 및 optional cell state c0은 다음 단일 레이어 네트워크의 출력이다.

$[h0 ; c0] = \tanh(W_z z + b_z)$

3.2 Sketch-RNN

HB

- 디코더의 각각의 단계에는 previous point S_{i-1}과 latent vector z를 concat한 x_i를 입력으로 준다.
- S_0은 (0,0,1,0,0)
- 각 단계의 출력은 다음 data point S_i의 확률분포이다.
- △x, △y는 GMM(Gaussian mixture model)을 사용해 다음 수식으로 나타낸다. P 는 (x,y의 상관관계를 뜻함)

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

- q1,q2,q3는 다음 data point의 상태가 각각 p1,p2,p3가 될 확률으로 q1+q2+q3=1이다.
- 따라서 out vector y는 5M+M+3의 parameter가 필요하다.

3.2 Sketch-RNN

HB

- RNN의 다음 hidden state는 forward operation을 통해 생성되며 fc-layer를 사용하여 output vector y_i 로 project된다.

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

- Vector y는 다음과 같은 parameter들로 구성된다.

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

- Exp 연산을 통해 표준 편차를 양수로 만들어주고 tanh연산을 통해 p값을 -1~1사이의 값으로 맞춘다.

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

- Categorical 분포의 확률은 출력을 logit values로 사용하여 계산된다.

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

3.2 Sketch-RNN

HB

- Key challenge는 모델이 언제 drawing을 멈추는지 학습하는 것.
- P1에 비해 p3 상태는 sketch당 한번 뿐이므로 압도적으로 적다.
- 이를 위해 p1, p2, p3에 적절한 가중치(1,10,100)를 곱하는 방식이 제안되었으나 적절하지 않다.
- 따라서 데이터 셋 중 가장 긴 sketch 데이터의 길이를 N_{max}로 설정 한 후 각 스케치의 길이(N_s)보다 큰 i번째 데이터에는 (0,0,0,0,1)를 넣어주어 균형을 맞춰준다.

3.2 Sketch-RNN

HB

- Train이 끝난 후 우리는 우리의 모델로부터 sketch를 sample할 수 있다.
- Sampling process동안 우리는 GMM과 categorical distribution parameter를 각 단계에서 생성하고 다음 단계를 위한 결과 s'_i를 생성한다.
- 인코더와 마찬가지로 샘플링된 출력은 deterministic한 출력이 아니라 input latent vectorz 에따라 조절되는 랜덤 시ㄱ퀀스이다.
- 이렇게 발생하는 랜덤성의 수준을 temperature parameter T 로 조절할 수 있다.

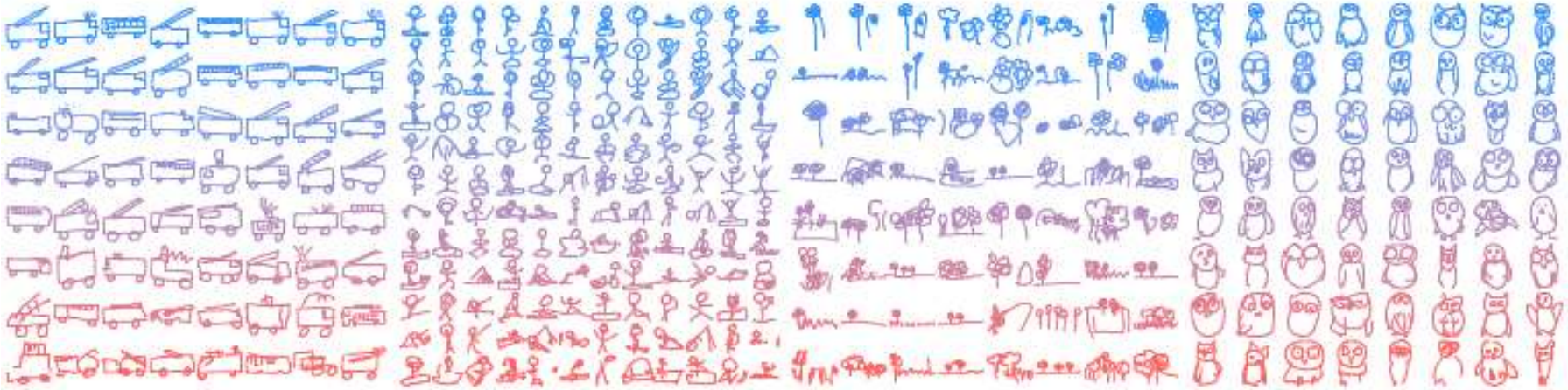
$$\hat{q}_k \to \frac{\hat{q}_k}{\tau},\ \hat{\Pi}_k \to \frac{\hat{\Pi}_k}{\tau},\ \sigma_x^2 \to \sigma_x^2\tau,\ \sigma_y^2 \to \sigma_y^2\tau. \qquad (8)$$

- T가 0에 가까워 질 수록 랜덤성이 줄어든다(T 은 0~1사이의 값)

3.3 Unconditional generation

HB

- 특별한 경우 입력이나 latent vector 없이 디코더 RNN 모듈만 train하는 모델을 학습하여 스케치를 unconditionally 하게 생성하도록 할 수 있다.
- 인코더를 제거하면 디코더 RNN은 latent vector가 없는 auto-regressive 모델이 된다.
- 아래는 unconditionally 하게 temperature parameter를 이용해 sampling한 결과



3.4 Training

HB

- Training procedure은 Variational Autoencode의 방법을 따른다.
- Loss function은 Reconstruction loss L_R, Kullback-Lebler Divergence Loss L_{KL}로 구성된다.

$$Loss = L_R + w_{KL} L_{KL}.$$

- 모델은 위 두 loss를 최적화 하는 방향으로 train.

3.4 Training

HB

- Reconstruction Loss

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

$$L_p = -\frac{1}{N_{\max}} \sum_{i=1}^{N_{\max}} \sum_{k=1}^{3} p_{k,i} \log(q_{k,i}), \; L_R = L_s + L_p. \tag{9}$$

- L_S는 각 stroke의 변위량에 대한, L_P는 각 stroke의 펜 상태에 대한 loss이다.
- L_S의 내부 값은 각 stroke의 변위에 대한 확률을 의미하며 이 값이 커진다는 것은 모델이 확실하게 stroke의 변위를 찾을 수 있다는 뜻이다.
- L_p역시 각 stroke의 확률을 의미하며 이 값이 커진다는 것은 모델이 확실하게 펜 상태를 결정할 수 있다는 것.
- 따라서 Loss function에서는 음수값을 붙여 위 확률이 커질수 있도록 train 한다.

- L_p식과 L_S식의 다른 점은 위에서 펜 상태의 균형을 위해 (0,0,0,0,1) vector를 넣어줬으므로 L_P는 N_{max}까지 계산을 한다는 점이다.

3.4 Training

HB

- Kullback-Lebler Divergence Loss

$$L_{KL} = -\frac{1}{2N_z}\left(1 + \hat{\sigma} - \mu^2 - \exp(\hat{\sigma})\right). \qquad (10)$$

- latent vector z의 분포와 평균이 0이고 unit variance를 갖는 IID 가우스 벡터간의 차이를 측정한다.

$$Loss = L_R + w_{KL} L_{KL}.$$

3.4 Training

HB

- 두 loss간의 trade off가 있다.

$$Loss = L_R + w_{KL} L_{KL}.$$

- W_{KL}을 0으로 설정 하면 더 나은 reconstruction loss를 얻는 대신 ability to enforce a prior over our latent space를 희생하는 pure autoencode에 가까워진다.

- Unconditional generation의 경우 우리의 모델은 standalone decode가 되므로 L_{KL}항은 없다.

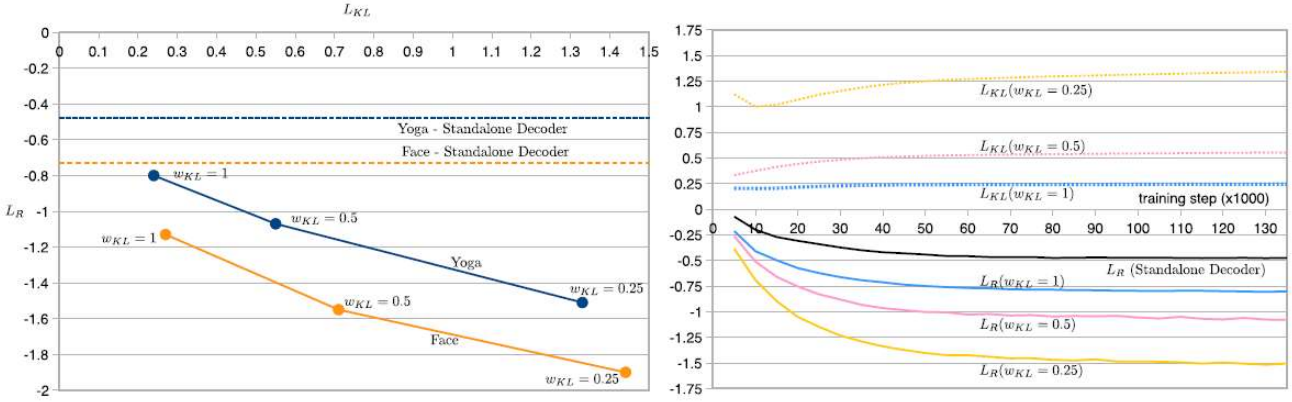


Figure 4: Tradeoff between L_R and L_{KL}, for two models trained on single class datasets (left). Validation Loss Graph for models trained on the Yoga dataset using various w_{KL} (right).

3.4 Training

HB

- 두 loss간의 trade off가 있다.

$$Loss = L_R + w_{KL} L_{KL}.$$

- W_{KL}을 0으로 설정 하면 더 나은 reconstruction loss를 얻는 대신 ability to enforce a prior over our latent space를 희생하는 pure autoencode에 가까워진다.

- Unconditional generation의 경우 우리의 모델은 standalone decode가 되므로 L_{KL}항은 없다.

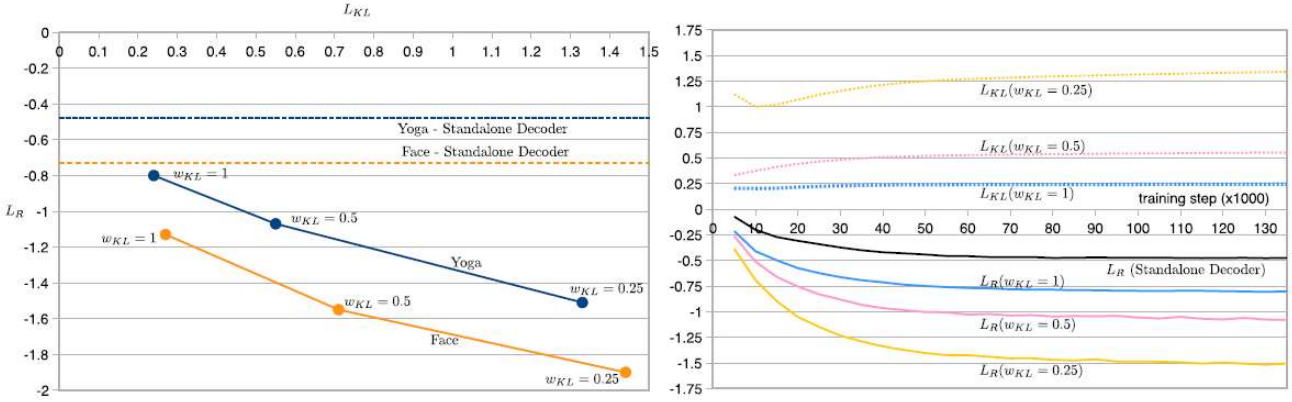


Figure 4: Tradeoff between L_R and L_{KL}, for two models trained on single class datasets (left). Validation Loss Graph for models trained on the Yoga dataset using various w_{KL} (right).

4. Experiments

HB

- Conditional and unconditional 벡터 이미지 생성을 위해 여러 실험을 진행한다.
- W_{KL}값을 조절해 가며 각각의 카테고리에 대한 Loss 값을 기록했다.

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

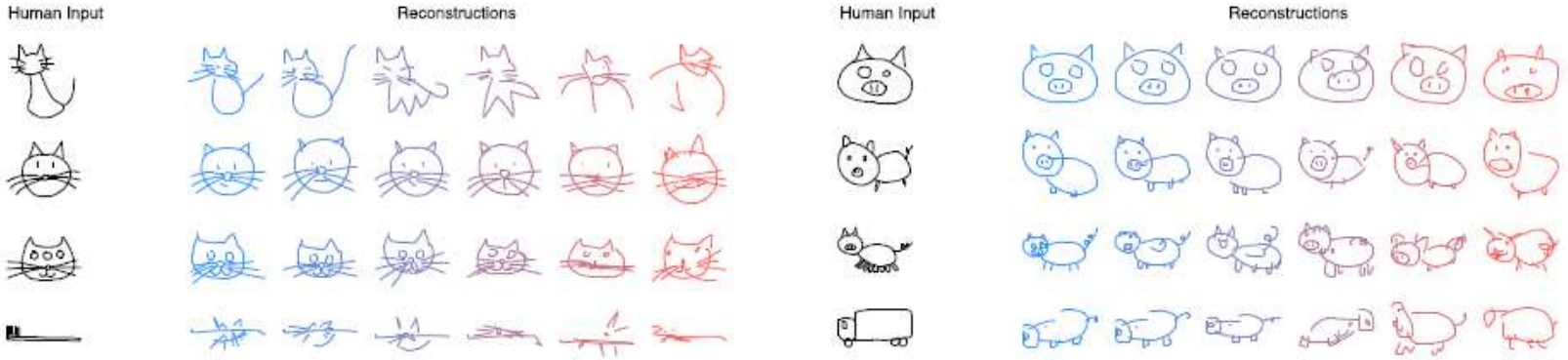
Table 1: Loss figures (L_R and L_{KL}) for various w_{KL} settings.

- 예상대로 W_{KL}값이 증가할 수록 L_R값이 감소하는 반면 L_{KL}값이 증가한다.

4.1 Conditional reconstruction

HB

- 입력 스케치 S가 주어져 재구성된 스케치 S'를 정성적으로 평가합니다.

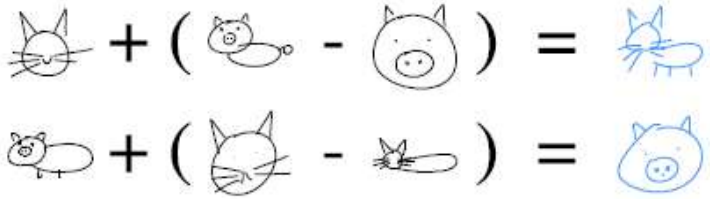


- 왼쪽 결과는 고양이 단일 클래스, 오른쪽 결과는 돼지 단일 클래스로 학습된 모델의 결과이다.
- 좌측(0,01)에서 오른쪽(1.0)까지 선형적으로 증가하는 T parameter에 대한 결과이다.
- 입력 이미지로 다른 물체의 스케치를 넣어도 각 클래스와 유사한 스케치를 생성한다.
- 재구성된 스케치는 입력 이미지와 유사한 특성을 가지며 때때로 세부 정보를 추가하거나 제거한다.

4.2 Latent space interpolation

HB

- 잠재 벡터 사이를 interpolate함으로써 한 영상이 다른 영상으로 어떻게 변형되는지 시각화 하였다.



- 높은 W_{KL}값을 사용하면 낮은 값을 사용한 모델과 비교하여 data manifold에 더 가까운 이미지를 생성할 것으로 예상한다.

- 위의 결과는 예상대로 더 높은 W_{KL}값으로 훈련된 모델이 보다 일관성 있는 보간 이미지를 생성하는 것을 보여준다.

4.3 Sketch drawing analogies

HB

- 위 결과의 interpolation 예는 latent vector z가 스케치의 개념적 특징을 인코딩 하는 것을 나타낸다.
- 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.
- 예를 들어 인코딩 된 돼지 머리의 letnet vector를 전체 돼지의 latent vector에서 빼서 돼지 몸통을 나타내는 vector를 얻을 수 있다.
- 이러한 drawing analogies를 통해 생성된 스케치의 manifold에서 다양한 개념을 표현하기 위해 latent space를 구성하는 방법을 탐색할 수 있다.

4.4 PREDICTING DIFFERENT ENDINGS OF INCOMPLETE SKETCHES

HB

- Sketch-RNN을 사용하여 미완성 sketch를 완성할 수 있다.
- 디코더 RNN을 standalone model로 사용하여 이전 data point에서 조건화된 스케치를 생성할 수 있다.
- 먼저, 디코더 RNN을 사용하여 미완성 sketch를 hidden state h로 인코딩한 후 h를 사용하여 나머지 sketch point를 생성한다

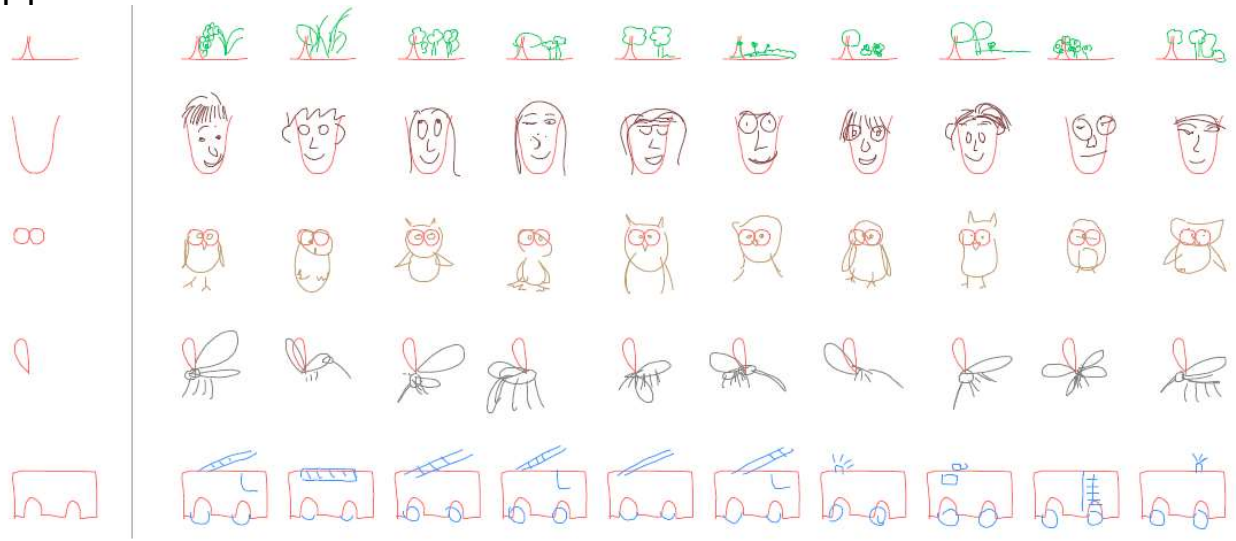


Figure 7: `sketch-rnn` predicting possible endings of various incomplete sketches (the red lines).

5. Applications and futer work

HB

- 다양한 클래스로 훈련된 디코더 전용 모델도 스케치를 완성하는 여러 가지 방법을 제시해 아티스트의 상상력 확장에 도움을 줄 수 있다.
- Conditional 모델에서 서로 다른 물체의 latent space를 탐색하는 것은 예술가들이 서로 다른 도면 사이의 흥미로운 관계를 찾을 수 있다.
- 패턴 디자이너는 sketch-RNN을 적용하여 유사하지만 독특한 다수의 디자인을 생성할 수 있다.
- 크리에이티브 디자이너가 대상 고객에게 더 큰 반향을 일으킬 수 있는 추상적 디자인을 고안하는 데 도움이 될 수 있는 애플리케이션으로 확장할 수 있다. (의자 같은 고양이)

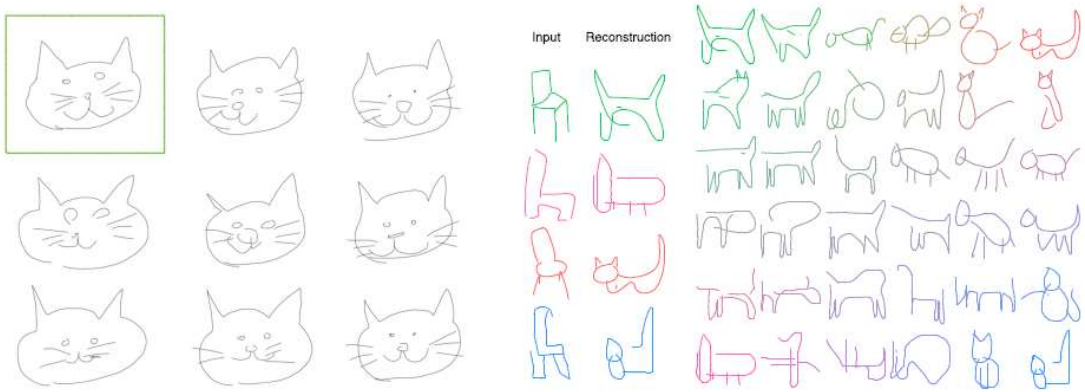


Figure 8: Generating similar, but unique sketches based on a single human sketch in the box (left). Latent space of generated cats conditioned on sketch drawings of chairs (right).

5. Applications and futer work

HB

- 고품질 스케치에 대해 훈련된 모델은 학생들에게 그림 그리는 법을 가르쳐 줄 수 있는 교육 application 에 사용 가능하다.
- 향후에는 train 과정에 사용자 등급 데이터를 통햅하여 sketch의 aesthetics를 최대화 할 수 있다.
- Sequence-Generation 모델의 hybrid 변형을 image-to-image 모델과 같은 unsupervised cross-domain pixel image generation models과 결합하는 것도 흥미로운 방향이다.
- 이 모델을 Pix2Pix와 같은 supervised cross-domain 모델과 결합하여 생성된 고양이 스케치에서 가끔 사실적인 고양이 이미지를 생성할 수 있을 것이다.

6. Conclusion

HB

- RNN을 사용하여 스케치 drawing을 모델링 하는 methodology를 개발한다.
- Sketch-RNN은 기존 스케치를 완료하는 가능한 방법을 생성할 수 있다.
- 우리의 모델은 또한 기존 스케치를 latent vector로 인코딩할 수 있으며 latent space 공간에 유사한 모양의 스케치를 생성할 수 있다.
- 두 스케치의 latent space 사이를 interpolate하여 무엇을 의미하는지 설명하고 속성을 조작할 수 있다.
- 스케치 drawing의 대규모 데이터 세트를 사용할 수 있게 됨으로써, generative vector image modelling 분야에서 추가 연구와 개발을 장려할 수 있기를 바란다.

7. Code

HB

- Reconstruction Loss

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

$$L_p = -\frac{1}{N_{\max}} \sum_{i=1}^{N_{\max}} \sum_{k=1}^{3} p_{k,i} \log(q_{k,i}), \; L_R = L_s + L_p.$$

```
def get_lossfunc(z_pi, z_mu1, z_mu2, z_sigma1, z_sigma2, z_corr,
                 z_pen_logits, x1_data, x2_data, pen_data):
  """Returns a loss fn based on eq #26 of http://arxiv.org/abs/1308.0850."""
  # This represents the L_R only (i.e. does not include the KL loss term).

  result0 = tf_2d_normal(x1_data, x2_data, z_mu1, z_mu2, z_sigma1, z_sigma2,
                         z_corr)
  epsilon = 1e-6
  # result1 is the loss wrt pen offset (L_s in equation 9 of
  # https://arxiv.org/pdf/1704.03477.pdf)
  result1 = tf.multiply(result0, z_pi)
  result1 = tf.reduce_sum(result1, 1, keep_dims=True)
  result1 = -tf.log(result1 + epsilon)  # avoid log(0)

  fs = 1.0 - pen_data[:, 2]  # use training data for this
  fs = tf.reshape(fs, [-1, 1])
  # Zero out loss terms beyond N_s, the last actual stroke
  result1 = tf.multiply(result1, fs)

  # result2: loss wrt pen state, (L_p in equation 9)
  result2 = tf.nn.softmax_cross_entropy_with_logits(
      labels=pen_data, logits=z_pen_logits)
  result2 = tf.reshape(result2, [-1, 1])
  if not self.hps.is_training:  # eval mode, mask eos columns
    result2 = tf.multiply(result2, fs)

  result = result1 + result2
  return result
```

7. Code

HB

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

```
# KL cost
self.kl_cost = -0.5 * tf.reduce_mean(
    (1 + self.presig - tf.square(self.mean) - tf.exp(self.presig)))
self.kl_cost = tf.maximum(self.kl_cost, self.hps.kl_tolerance)
```

7. Code

HB

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

```
if hps.conditional:  # vae mode:
  self.mean, self.presig = self.encoder(self.output_x,
                                         self.sequence_lengths)
  self.sigma = tf.exp(self.presig / 2.0)  # sigma > 0. div 2.0 -> sqrt.
  eps = tf.random_normal(
      (self.hps.batch_size, self.hps.z_size), 0.0, 1.0, dtype=tf.float32)
  self.batch_z = self.mean + tf.multiply(self.sigma, eps)
```