**Lab 5: Conditional Sequence-to-sequence VAE**

**ID: 0856601**

**Name: 黃兆宇**

**1. Introduction**

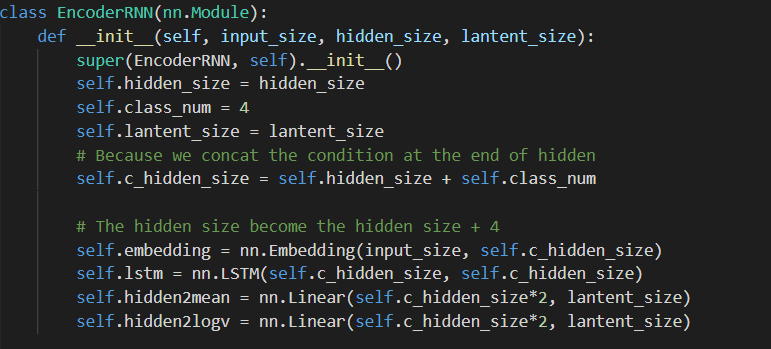
In this homework, we need to finish the English tense conversion and word generation task. We are going to use conditional VAE (conditional variational auto-encoder) to build a sequence to sequence model, the training data contains 1227 training pairs, and the test data contains 10 testing pairs, in test data, the tense is different for the words in testing pairs. For the tense conversion task, the input of encoder is the input test word and its tense, and the input of decoder is the target tense we want to convert and a lantent vector sampled from the output distribution of encoder, and the output of the model should be the input word with the target tense. For the generation task, the input of model should be the a random lantent vector, and the output should be a word with target tense.

2. Derivation of CVAE

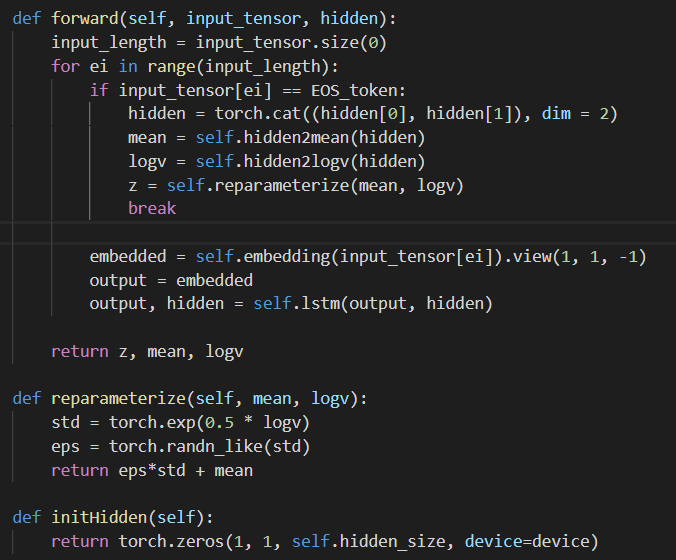
3. Implementation details

3.A. Model implementation

3.A-1. Encoder



For the encoder part of the model, the \_\_init\_\_() function initialize the layers which will be applied in the forward() function. Most of layers is same as the setting in normal RNN. And to implement VAE, two layers are added, including hidden2mean() and hidden2logv().

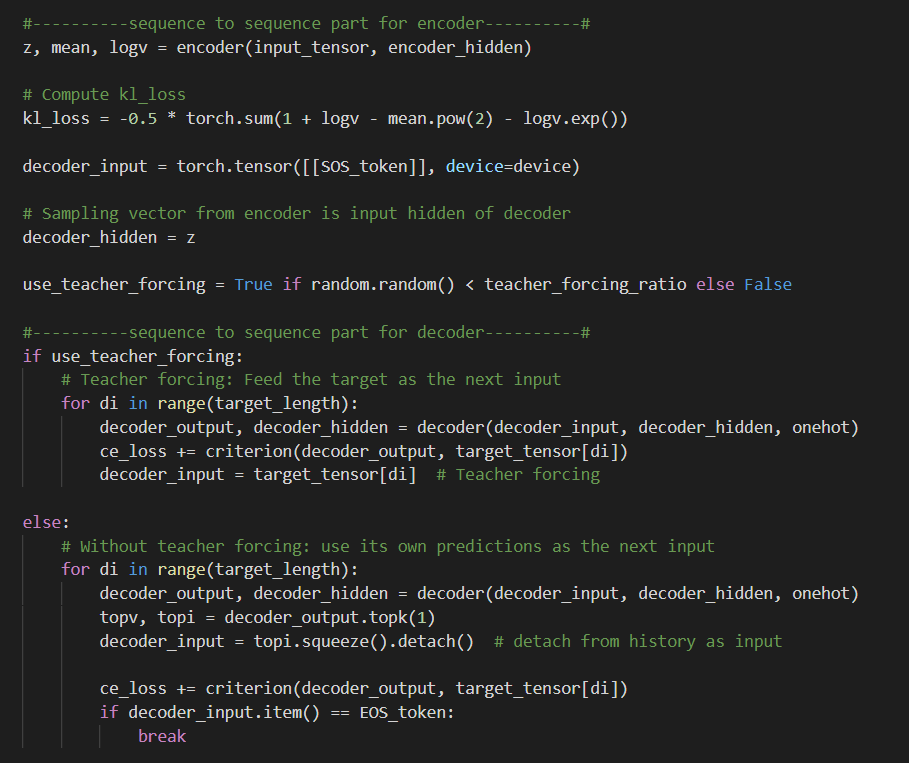


In the forward() function of the encoder, the structured is a little different with the normal RNN, instead of input one character (ex. The id of ‘e’) to the model and output the hidden to the next time step’s encoder, I

put the whole word to the model (ex. A vector of ids corresponding to the ‘e, x, p, e, n, d’).

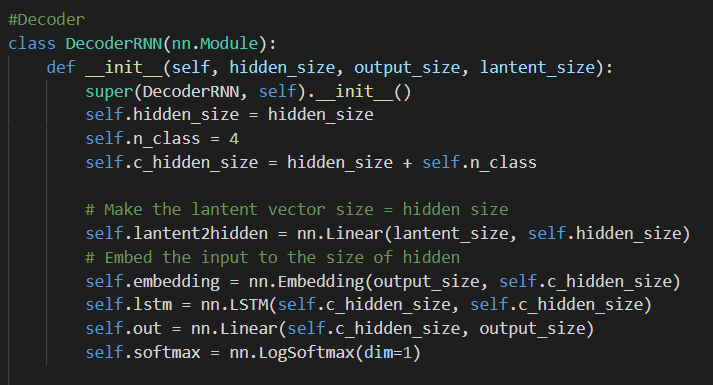
And when taking the EOS token, the cell state (hidden[0]) and hidden state (hidden[1]) will be cat together and sent to two linear layers. One of the linear layers will output a mean value and another will output log variance. These two values represent the output Gaussian distribution of the VAE.

And the reparameterization trick is applied, a pytorch function randn\_like will sample a vector(eps) with size std from Normal distribution. And the z vector is compute by eps\*std + mean.

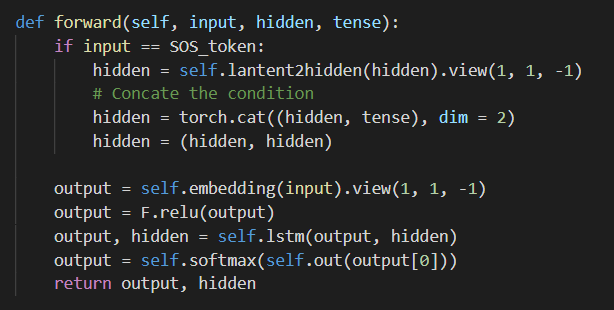


The decoder’s input (decoder\_hidden) is sampled from a Gaussian distribution created by encoder.

3.A-2. Decoder

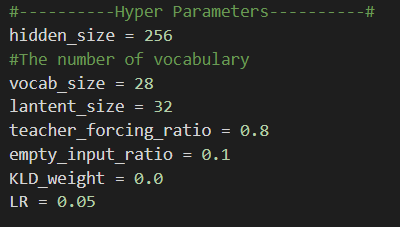


In the \_\_init\_\_() function of the decoder, most of the layers are similar with normal RNN.



And because of condition of the word will be considered in this lab, an onehot vector will be sent to the decoder and cat with the lantent code which is output from the encoder.

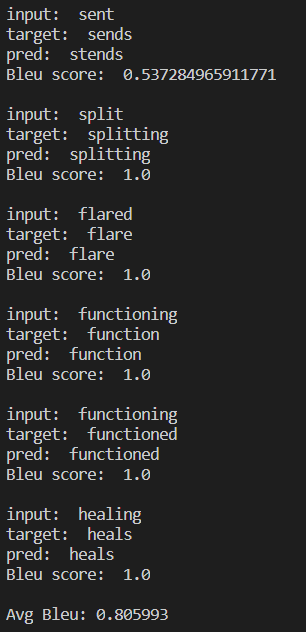
3-B Hyperparameters



Above picture is the hyperparameters I used in this lab, vocab size is the character number: 26 + SOS, EOS:2, lantent size is the size of input vector(z) to the decoder, teacher\_forcing\_ratio is the probability that the decoder use teacher forcing, KLD\_weigt(KL cost) are the initial KL-Divergence weight for training and LR is learning rate.

4. Results and discussion

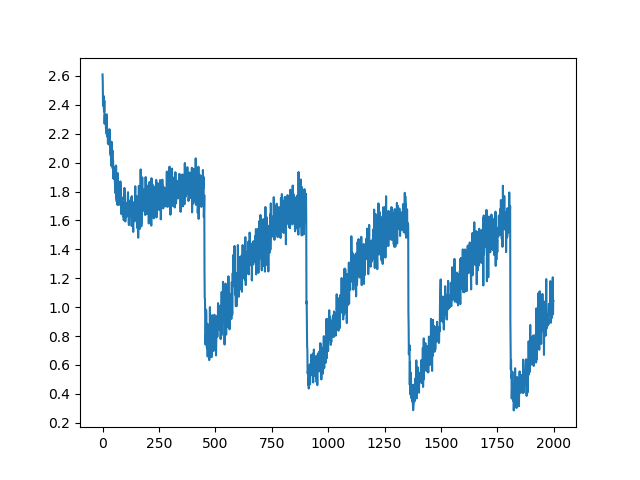
4.A Experiment result:

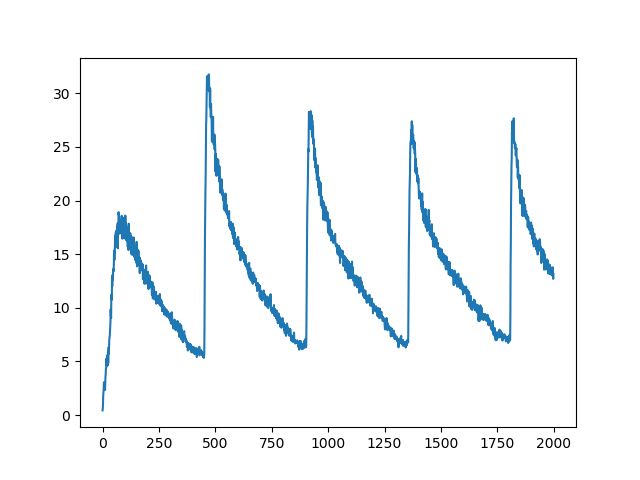
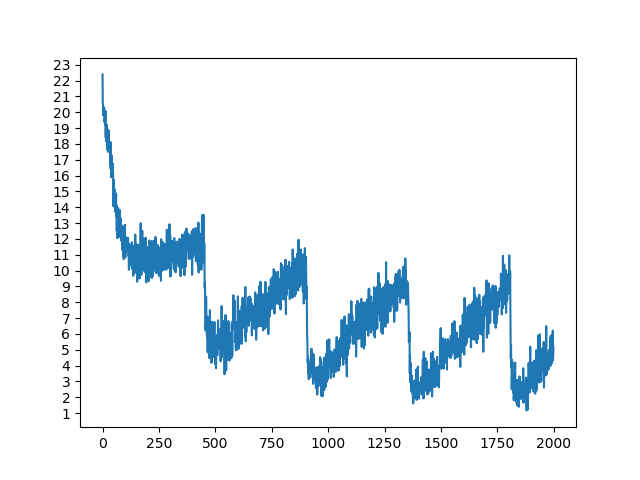


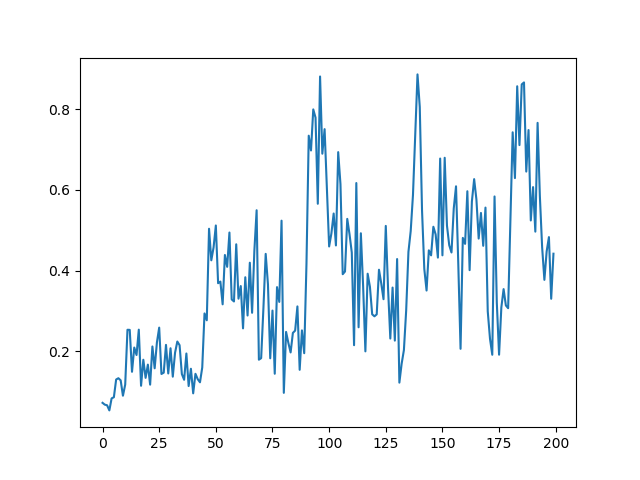
The best score I get for tense conversion is 0.805993. (Encoder, Decoder training for 185000 epoches)



The best score I get for generation task is 0.23. (Decoder training for 135000 epoches)

Total Loss (Cross Entropy + KLD)

Cross Entropy Loss KL-divergence loss

Bleu score

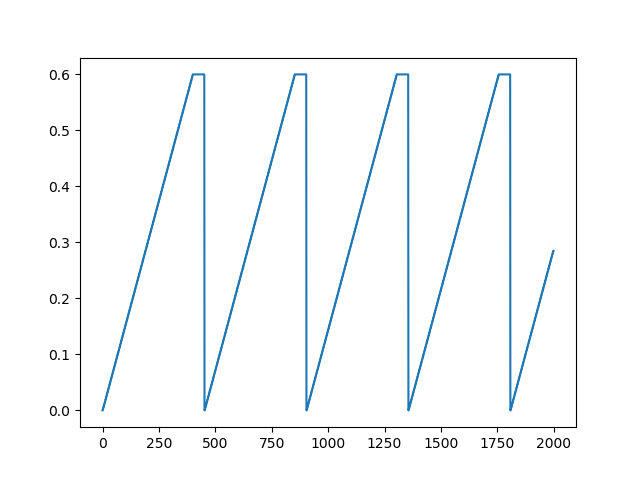
Unit: 100 iters

4.B Result discussion:

In this Lab, I tried both cyclic and monotonic KL cost annealing and find out that the cyclic method can achieve higher bleu score. I only tried two kinds of teacher forcing rate: 0.8 and 1.0, but I didn’t find the big difference of the output at these two different teacher forcing rate, so I focus on adjusting the KL cost in this lab during training to get a better result. And for learning rate, I choose to keep the initial setting (0.05) in the sample code.

4.B-1. Cyclic annealing Result (Best Result)

Cyclic KL cost annealing



Unit: 100 iters

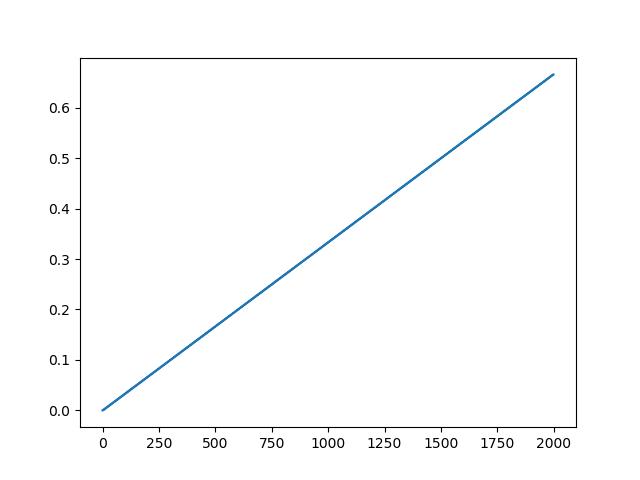
The KL cost (kld\_weight\_update) will increase 0.003 every 200 iters (words), and when the KL cost achieve 0.6, the KL cost will be hold for 5000 iters, and then the KL cost will be set to 0, and then the next cycle will start.

By observing the loss changing trend in 4-A, the cyclic method first will let the model focus on its cross-entropy loss and when the cross entropy loss go to a low point, the model will start to reduce its kl-divergence loss, and we can also observe that the curve of cross-entropy loss and KL-divergence loss go to opposite directions at the same time interval, which means that the model cannot reduce both of them simultaneously.

By evaluating the test dataset during training, I also find out that reducing cross-entropy loss will increase the reconstruction result for the input words, which means the output for the decoder will be similar with decoder. When increase the KL cost, the model will start to reduce the KL cost, in this phase, the decoder will focus on output the word that fits the condition tense.

And by observe the bleu score, I find out the score will go to a peak after the KL cost is set to 0, and the peak will increase in the first few cycles. And the best model can be chosen by observing these peak value.

4.B-2. Monotonic annealing Result

Monotonic KL cost annealing

For the monotonic cost annealing, I increase the KL cost 0.01 every 3000 iters, but I didn’t get the bleu score as high as the cyclic method. And I also observe that although the KL-divergence loss keep go down, but the cross entropy loss and total loss will rise up which produce a higher loss then the cyclic method.