DLP HW5

309553008

- Introduction:

  這次功課是在實作一個 seq2seq 的 VAE，input data 為動詞與其四種型態(簡單式、第三人稱單數、現在進行式、過去式)，將型態當成 condition 與動詞一起送進 Encoder 壓成 latent code，然後將 latent code 與 target 的 condition concat 在一起形成 Decoder 的 hidden，然後將’SOS’的 token 與 hidden 當成最開始 Decoder 的 input，將產出的 output、hidden、cell 再當 input 丟進 Decoder 直到 Max length 或者’EOS’ token，並將每一輪的 output 中機率最高的那個字拿出來組成一串詞，最後與 target 比，更新 Loss。

- Derivation of CVAE:

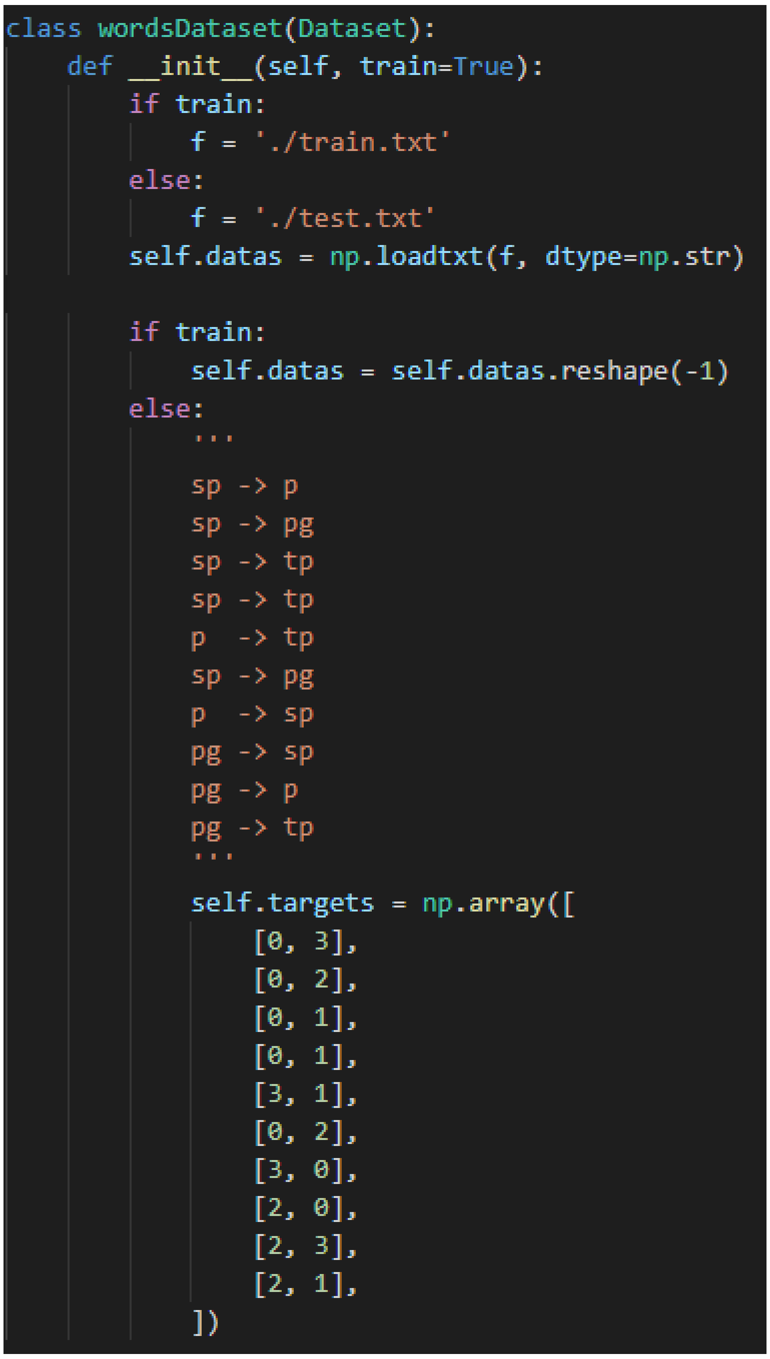
$$\underbrace{E_{\boldsymbol{Z}\sim q(\boldsymbol{Z}|\boldsymbol{X};\boldsymbol{\theta}')}\log p(\boldsymbol{X}|\boldsymbol{Z};\boldsymbol{\theta})}_{\text{Reconstruction}}\underbrace{-\mathrm{KL}(q(\boldsymbol{Z}|\boldsymbol{X};\boldsymbol{\theta}')||p(\boldsymbol{Z}))}_{\text{Regularization}}$$

  多一個已知條件 C (condition)，也就是說在 Encoder 與 Decoder 的條件機率後面會多一個 C

$$E_{\boldsymbol{Z}\sim q(\boldsymbol{Z}|\boldsymbol{X},c;\boldsymbol{\theta}')}\log p(\boldsymbol{X}|\boldsymbol{Z},c;\boldsymbol{\theta})-\mathrm{KL}(q(\boldsymbol{Z}|\boldsymbol{X},c;\boldsymbol{\theta}')||\underbrace{p(\boldsymbol{Z}|c)})$$

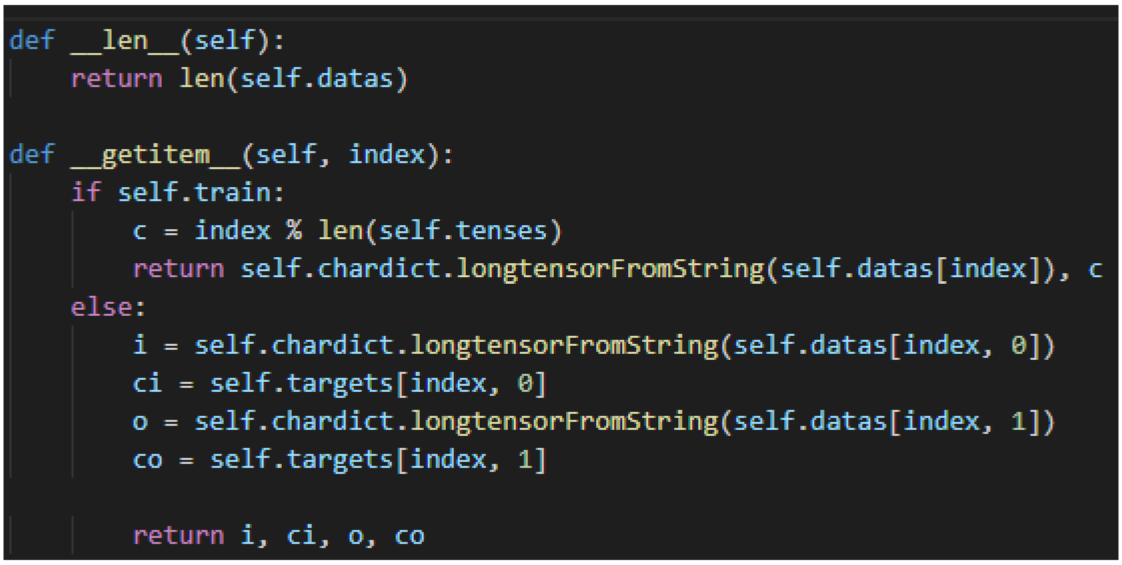
- Implementation details:
  1. Data Loader

     將檔案讀進來，若是 train 就攤平，若是 test 就按照 TA 給的 read me 裡的 Hint 把 target 的 condition 建好



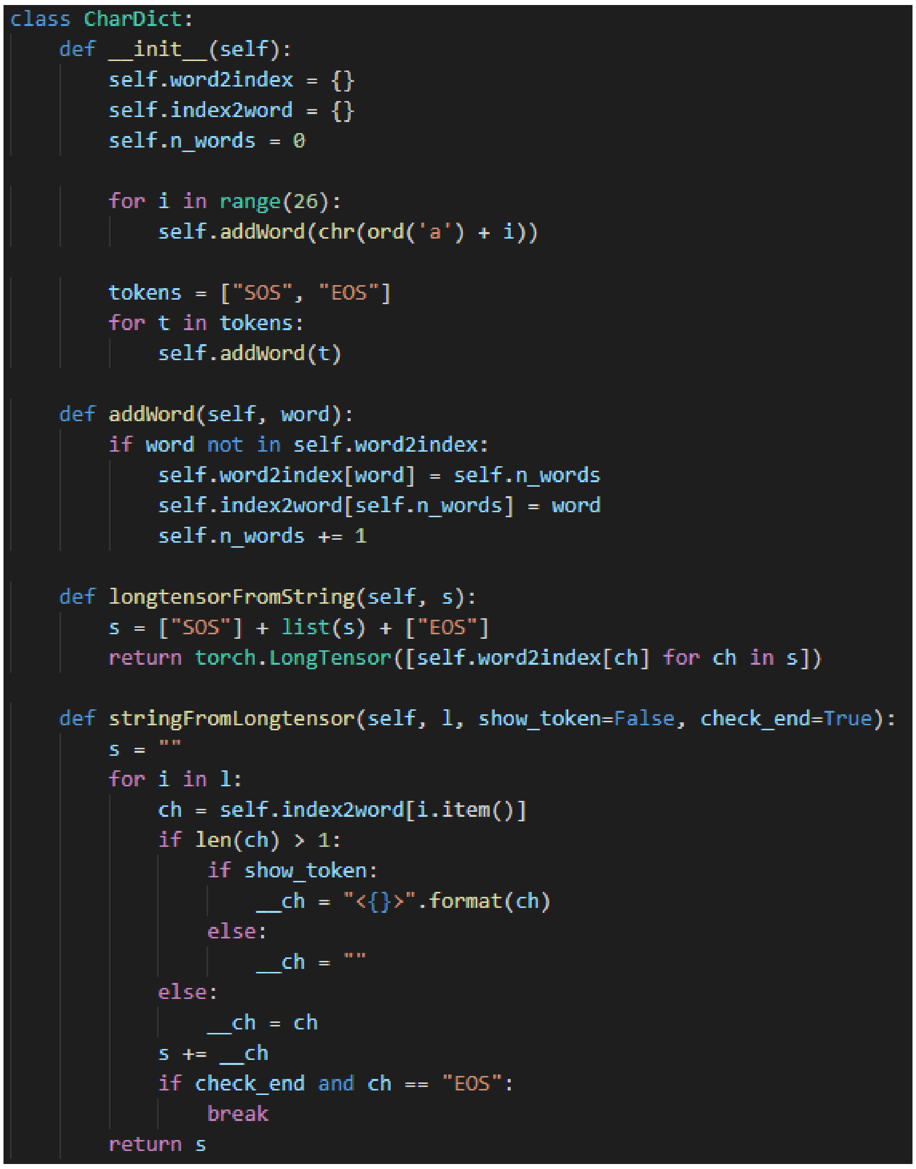
```
class wordsDataset(Dataset):
    def __init__(self, train=True):
        if train:
            f = './train.txt'
        else:
            f = './test.txt'
        self.datas = np.loadtxt(f, dtype=np.str)

        if train:
            self.datas = self.datas.reshape(-1)
        else:
            '''
            sp -> p
            sp -> pg
            sp -> tp
            sp -> tp
            p  -> tp
            sp -> pg
            p  -> sp
            pg -> sp
            pg -> p
            pg -> tp
            '''
            self.targets = np.array([
                [0, 3],
                [0, 2],
                [0, 1],
                [0, 1],
                [3, 1],
                [0, 2],
                [3, 0],
                [2, 0],
                [2, 3],
                [2, 1],
            ])
```

在 getitem 的部分，若是 train 就吐[‘SOS’token + 詞 + ‘EOS’token] 的 ASCII code 的 tensor 與該詞的 condition，若是 test 就吐 [‘SOS’token + origin word + ‘EOS’token]、origin word’s condition、[‘SOS’token + target word + ‘EOS’token]、target word’s condition



```python
def __len__(self):
    return len(self.datas)

def __getitem__(self, index):
    if self.train:
        c = index % len(self.tenses)
        return self.chardict.longtensorFromString(self.datas[index]), c
    else:
        i = self.chardict.longtensorFromString(self.datas[index, 0])
        ci = self.targets[index, 0]
        o = self.chardict.longtensorFromString(self.datas[index, 1])
        co = self.targets[index, 1]

        return i, ci, o, co
```

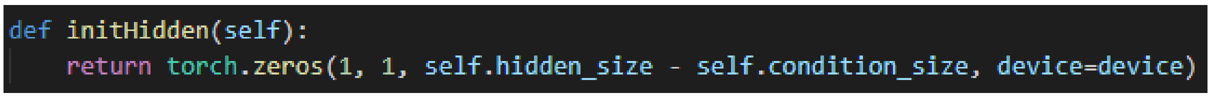
這邊作詞與 ASCII code 的轉換、並把前後都接上 token



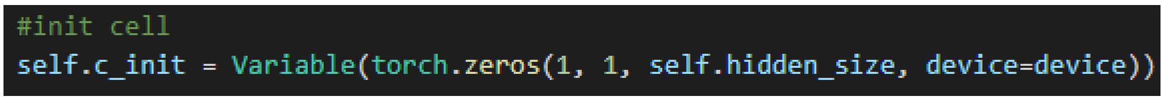
```python
class CharDict:
    def __init__(self):
        self.word2index = {}
        self.index2word = {}
        self.n_words = 0

        for i in range(26):
            self.addWord(chr(ord('a') + i))

        tokens = ["SOS", "EOS"]
        for t in tokens:
            self.addWord(t)

    def addWord(self, word):
        if word not in self.word2index:
            self.word2index[word] = self.n_words
            self.index2word[self.n_words] = word
            self.n_words += 1

    def longtensorFromString(self, s):
        s = ["SOS"] + list(s) + ["EOS"]
        return torch.LongTensor([self.word2index[ch] for ch in s])

    def stringFromLongtensor(self, l, show_token=False, check_end=True):
        s = ""
        for i in l:
            ch = self.index2word[i.item()]
            if len(ch) > 1:
                if show_token:
                    __ch = "<{}>".format(ch)
                else:
                    __ch = ""
            else:
                __ch = ch
            s += __ch
            if check_end and ch == "EOS":
                break
        return s
```

2. Encoder

先將 hidden 與 cell 做 initial

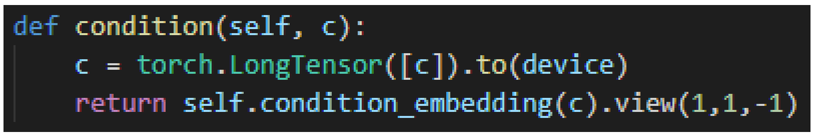


```
def initHidden(self):
    return torch.zeros(1, 1, self.hidden_size - self.condition_size, device=device)
```



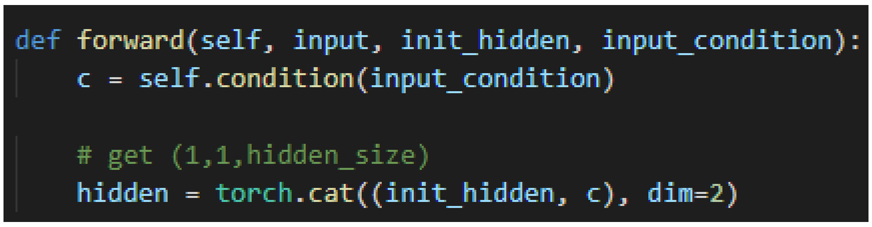
```
#init cell
self.c_init = Variable(torch.zeros(1, 1, self.hidden_size, device=device))
```

然後將 condition 做 embedding



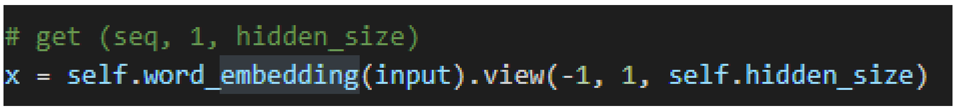
```
def condition(self, c):
    c = torch.LongTensor([c]).to(device)
    return self.condition_embedding(c).view(1,1,-1)
```

把 embedding 完的 condition cat 到 hidden 後面



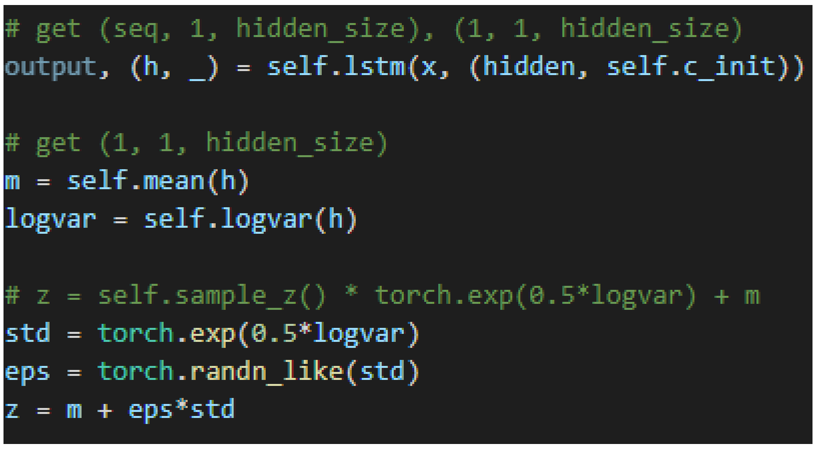
```
def forward(self, input, init_hidden, input_condition):
    c = self.condition(input_condition)

    # get (1,1,hidden_size)
    hidden = torch.cat((init_hidden, c), dim=2)
```

在來把 input 做 embedding



```
# get (seq, 1, hidden_size)
x = self.word_embedding(input).view(-1, 1, self.hidden_size)
```

最後丟到 LSTM，拿到更新後的 hidden，分別丟到兩個 FC 算 mean 跟 logvar，算完後用 mean 跟 logvar 去 sample (reparameterization trick)

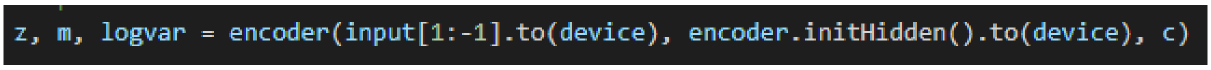


```
# get (seq, 1, hidden_size), (1, 1, hidden_size)
output, (h, _) = self.lstm(x, (hidden, self.c_init))

# get (1, 1, hidden_size)
m = self.mean(h)
logvar = self.logvar(h)

# z = self.sample_z() * torch.exp(0.5*logvar) + m
std = torch.exp(0.5*logvar)
eps = torch.randn_like(std)
z = m + eps*std
```

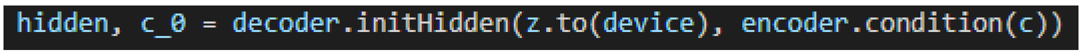
3. Decoder

將 Encoder sample 出來的 z 與 input 的 condition concat 起來，然後

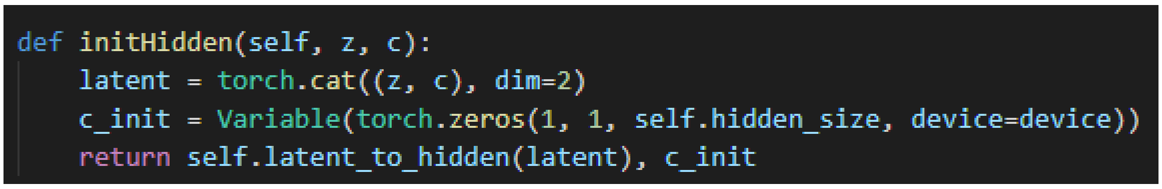
丟進 FC 裡轉成 hidden size



```
z, m, logvar = encoder(input[1:-1].to(device), encoder.initHidden().to(device), c)
```

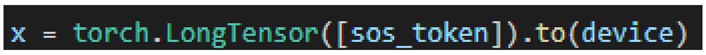


```
hidden, c_0 = decoder.initHidden(z.to(device), encoder.condition(c))
```



```
def initHidden(self, z, c):
    latent = torch.cat((z, c), dim=2)
    c_init = Variable(torch.zeros(1, 1, self.hidden_size, device=device))
    return self.latent_to_hidden(latent), c_init
```

Encoder 第一個 input 為'SOS' token



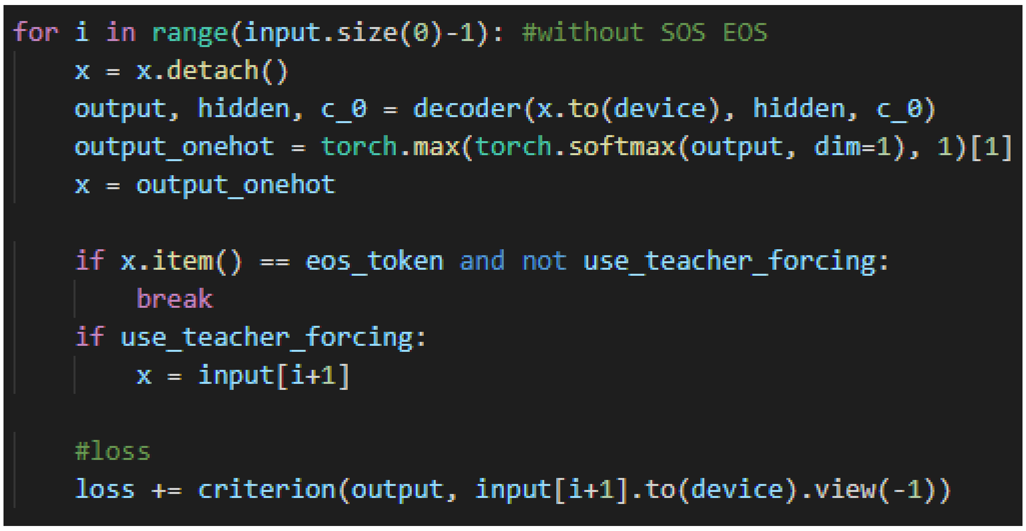
```
x = torch.LongTensor([sos_token]).to(device)
```

然後將 input 與 initial 後的 hidden 丟給 Decoder，將 output 中機率

最大的字(char)與更新後的 hidden、cell 當下個 input 一直做到

output 出'EOS' token 或當下詞的最大長度，最後將 Decoder 生出

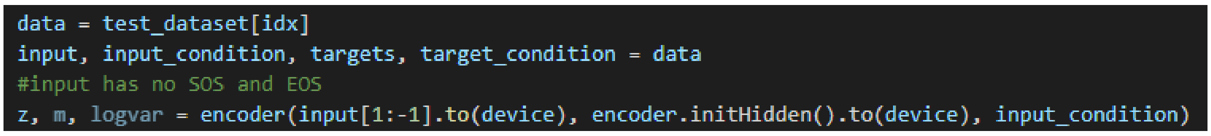
的詞與原本的詞做 Loss 更新



```
for i in range(input.size(0)-1): #without SOS EOS
    x = x.detach()
    output, hidden, c_0 = decoder(x.to(device), hidden, c_0)
    output_onehot = torch.max(torch.softmax(output, dim=1), 1)[1]
    x = output_onehot

    if x.item() == eos_token and not use_teacher_forcing:
        break
    if use_teacher_forcing:
        x = input[i+1]

    #loss
    loss += criterion(output, input[i+1].to(device).view(-1))
```

4. Evaluation

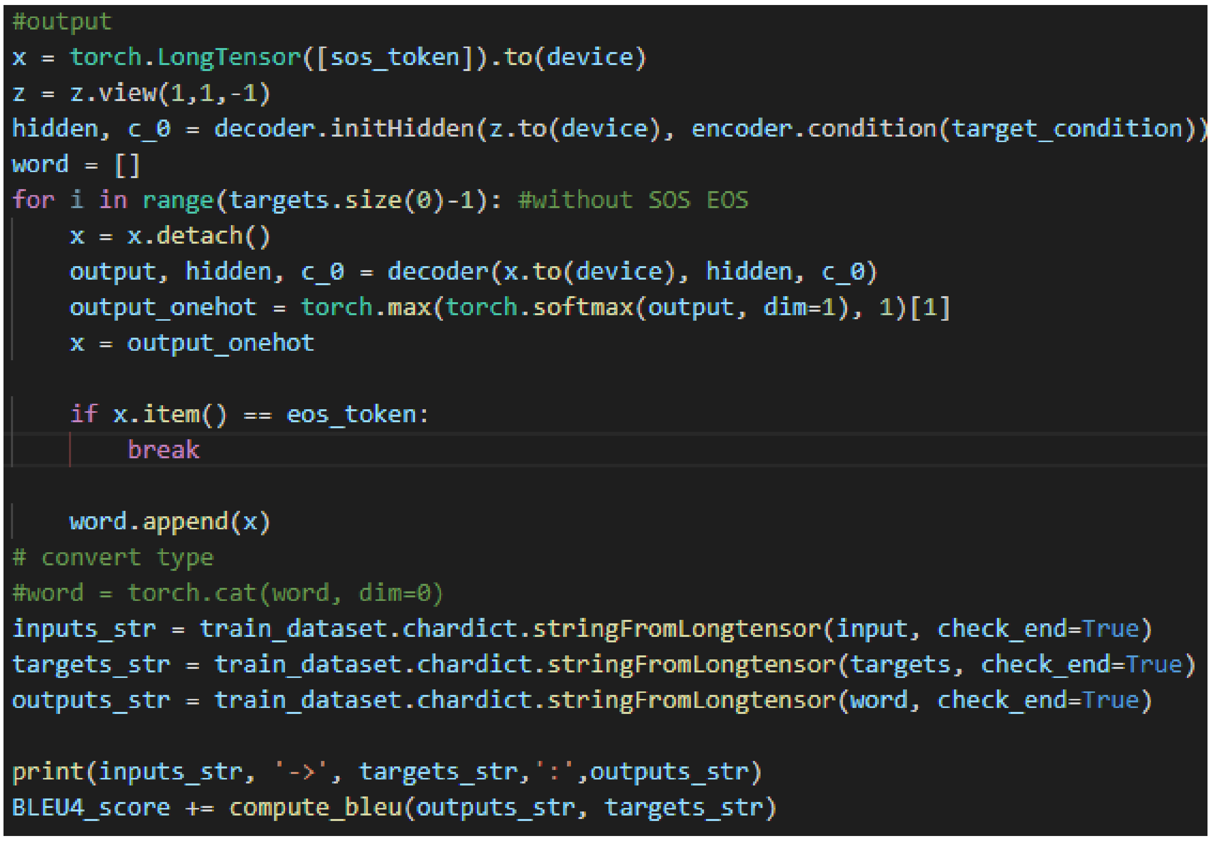
將 input 的詞與 condition 壓成 latent code



```python
data = test_dataset[idx]
input, input_condition, targets, target_condition = data
#input has no SOS and EOS
z, m, logvar = encoder(input[1:-1].to(device), encoder.initHidden().to(device), input_condition)
```

將 latent code 與 target condition concat 在一起去做 decode，最後

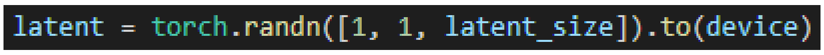
去計算 BLEU4 score



```python
#output
x = torch.LongTensor([sos_token]).to(device)
z = z.view(1,1,-1)
hidden, c_0 = decoder.initHidden(z.to(device), encoder.condition(target_condition))
word = []
for i in range(targets.size(0)-1): #without SOS EOS
    x = x.detach()
    output, hidden, c_0 = decoder(x.to(device), hidden, c_0)
    output_onehot = torch.max(torch.softmax(output, dim=1), 1)[1]
    x = output_onehot

    if x.item() == eos_token:
        break

    word.append(x)
# convert type
#word = torch.cat(word, dim=0)
inputs_str = train_dataset.chardict.stringFromLongtensor(input, check_end=True)
targets_str = train_dataset.chardict.stringFromLongtensor(targets, check_end=True)
outputs_str = train_dataset.chardict.stringFromLongtensor(word, check_end=True)

print(inputs_str, '->', targets_str,':',outputs_str)
BLEU4_score += compute_bleu(outputs_str, targets_str)
```

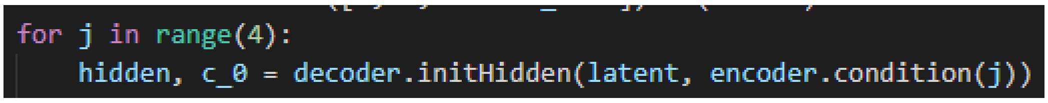
Generation:

從 Gaussian noise 中抽出一個 latent code 當 decoder 的 z



```python
latent = torch.randn([1, 1, latent_size]).to(device)
```

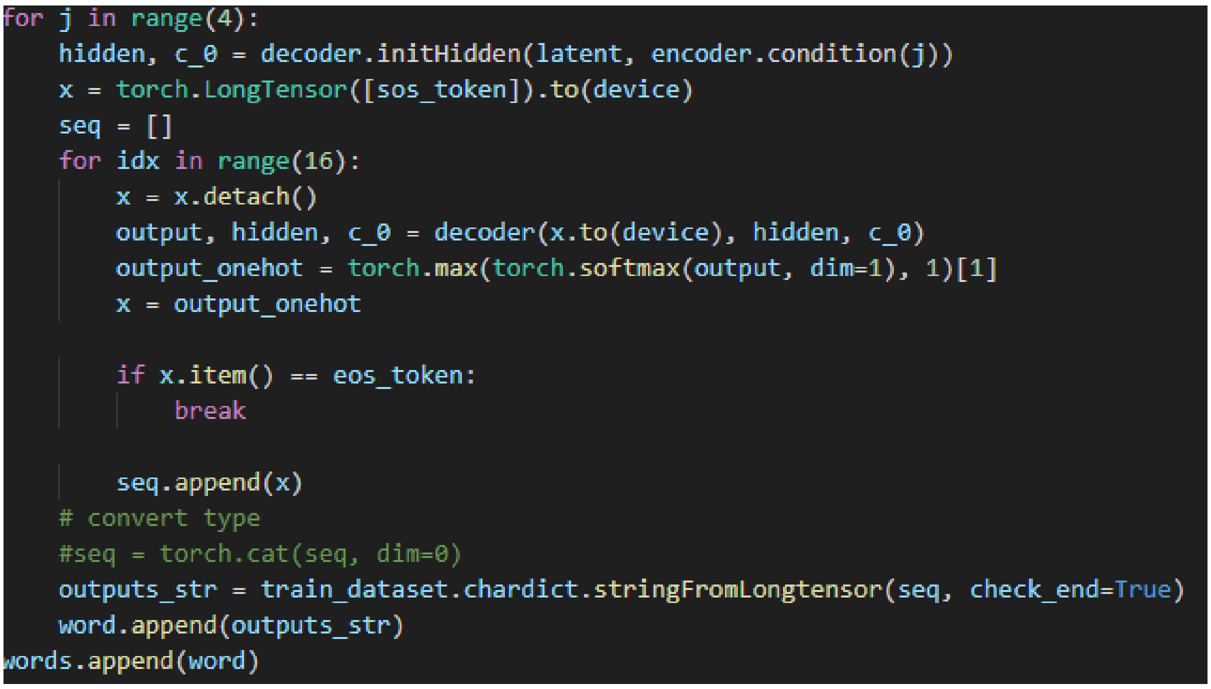
將 z 與 4 種詞態做 concat 當成 decoder 的 input



```python
for j in range(4):
    hidden, c_0 = decoder.initHidden(latent, encoder.condition(j))
```

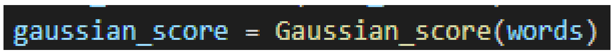
我設 max length 為 16，所以 Decoder 會一直生出字直到 16 個 char

或遇到’EOS’ token



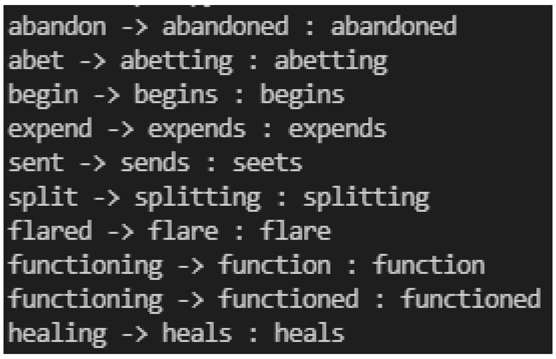
```
for j in range(4):
    hidden, c_0 = decoder.initHidden(latent, encoder.condition(j))
    x = torch.LongTensor([sos_token]).to(device)
    seq = []
    for idx in range(16):
        x = x.detach()
        output, hidden, c_0 = decoder(x.to(device), hidden, c_0)
        output_onehot = torch.max(torch.softmax(output, dim=1), 1)[1]
        x = output_onehot

        if x.item() == eos_token:
            break

        seq.append(x)
    # convert type
    #seq = torch.cat(seq, dim=0)
    outputs_str = train_dataset.chardict.stringFromLongtensor(seq, check_end=True)
    word.append(outputs_str)
words.append(word)
```

最後計算 Gaussian score



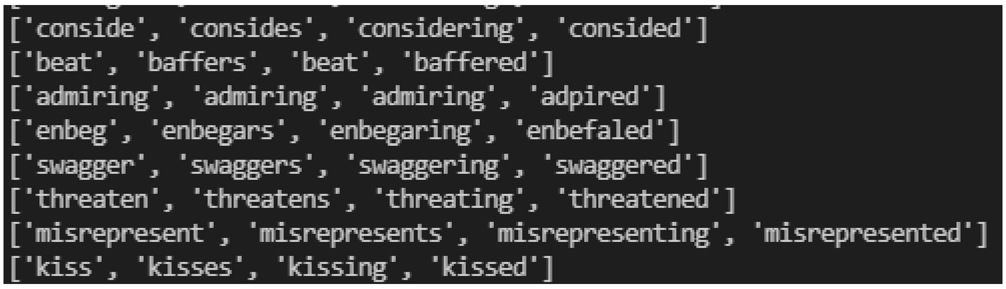
```
gaussian_score = Gaussian_score(words)
```

Test data:



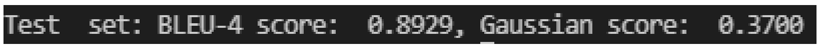
```
abandon -> abandoned : abandoned
abet -> abetting : abetting
begin -> begins : begins
expend -> expends : expends
sent -> sends : seets
split -> splitting : splitting
flared -> flare : flare
functioning -> function : function
functioning -> functioned : functioned
healing -> heals : heals
```

Generation output(部分):



```
['conside', 'condides', 'considering', 'consided']
['beat', 'baffers', 'beat', 'baffered']
['admiring', 'admiring', 'admiring', 'adpired']
['enbeg', 'enbegars', 'enbegaring', 'enbefaled']
['swagger', 'swaggers', 'swaggering', 'swaggered']
['threaten', 'threatens', 'threating', 'threatened']
['misrepresent', 'misrepresents', 'misrepresenting', 'misrepresented']
['kiss', 'kisses', 'kissing', 'kissed']
```

BLEU4 score, Gaussian score (best):

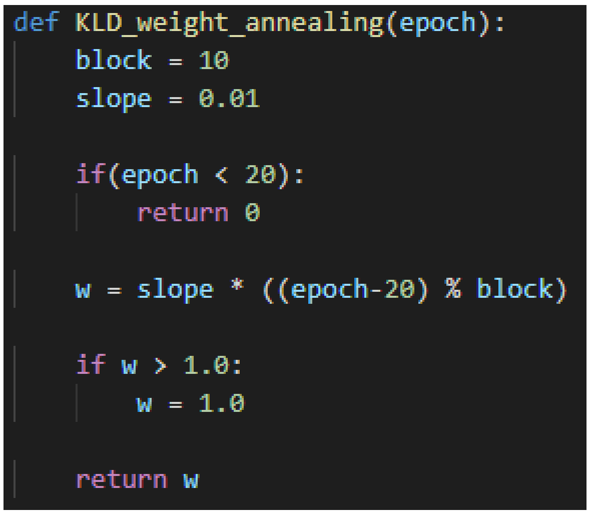


```
Test  set: BLEU-4 score:  0.8929, Gaussian score:  0.3700
```

Specify the hyperparameters:

a. KL weight:

   20 個 epoch 都是 0，之後以 0.01 的斜率上升(10 個 epoch 上升一次)

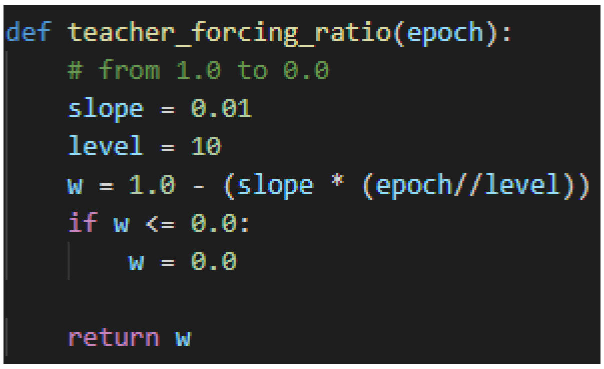


```python
def KLD_weight_annealing(epoch):
    block = 10
    slope = 0.01

    if(epoch < 20):
        return 0

    w = slope * ((epoch-20) % block)

    if w > 1.0:
        w = 1.0

    return w
```

b. learning rate : 0.007 (TA 給的)

c. teacher forcing ratio:

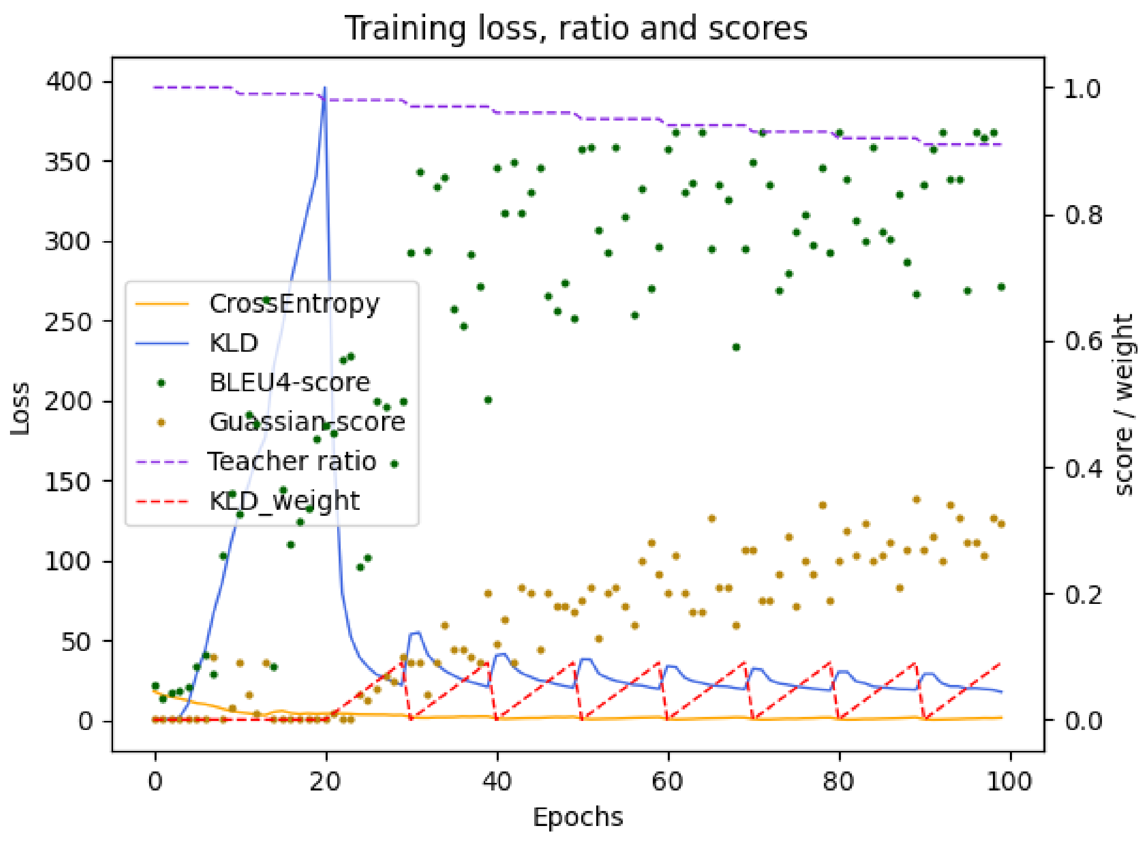
   從 0 上升到 1，10 個 epoch 下降一次



```python
def teacher_forcing_ratio(epoch):
    # from 1.0 to 0.0
    slope = 0.01
    level = 10
    w = 1.0 - (slope * (epoch//level))
    if w <= 0.0:
        w = 0.0

    return w
```

d. epochs : 100

- Results and discussion:



teacher forcing ratio 跟前述所說 10 個 epoch 下降一次，後來看了一下助教的圖，發現前面好幾個 epoch 都是 1，但是我覺得我現在的 performance 還不錯就沒有改了。

KL weight 我本來也是從 0 開始，10 個 epoch 上升一次，然後不只 BLEU4 score 爛到不行，Gaussian score 從來沒有 0 以外的結果，看完助教的圖後，我把前 20 的 epoch 都調 0，讓 KLD Loss 先往上飆後再降下來，才有比較好的結果，我推測應該是一開始 KLD Loss 太高，所以 CrossEntropy Loss 就沒有參考價值，這樣會導致 Decoder 沒有 train 到，整個 VAE 就爛掉了。