Hw2 Report b03705027 鄭從德

語言：Keras

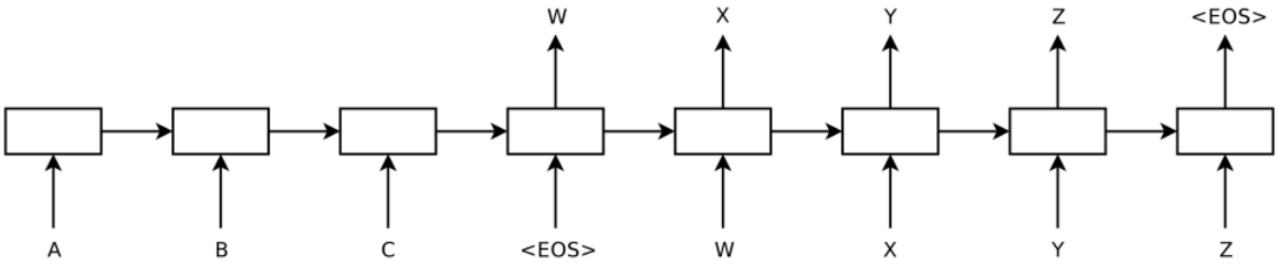
（一） Describe your seq2seq model:

這是我最終的 seq2seq model 結構。

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 80, 4096)	0	
lstm_1 (LSTM)	[(None, 80, 256), (No	4457472	input_1[0][0]
permute_1 (Permute)	(None, 256, 80)	0	lstm_1[0][0]
dense_1 (Dense)	(None, 256, 39)	3159	permute_1[0][0]
permute_2 (Permute)	(None, 39, 256)	0	dense_1[0][0]
dense_2 (Dense)	(None, 39, 256)	65792	permute_2[0][0]
concatenate_1 (Concatenate)	(None, 39, 512)	0	dense_2[0][0] permute_2[0][0]
input_2 (InputLayer)	(None, 39, 2300)	0	
dense_3 (Dense)	(None, 39, 256)	131328	concatenate_1[0][0]
lstm_2 (LSTM)	(None, 39, 256)	2618368	input_2[0][0] lstm_1[0][1] lstm_1[0][2]
concatenate_2 (Concatenate)	(None, 39, 512)	0	dense_3[0][0] lstm_2[0][0]
dense_4 (Dense)	(None, 39, 2300)	1179900	concatenate_2[0][0]

Total params: 8,456,019
Trainable params: 8,456,019
Non-trainable params: 0

其中 input_1 一直到我的 concatenate_1 是我的 encoder 以及 attention，而後面 input_2 到最後面是我的 decoder。他是如下這種最簡單 seq2seq 的變形：input_1 就是左半邊 encoder 吃個 video input，右半邊則是吃一串 label，進行 predict 下一個字。



前面的 encoder，再做完 LSTM 之後，我會把最後一個 states 傳給 decoder 當作 initial states，所以可以算其實是接成一個很長的 LSTM。但是除了傳 state 之外，我還做了 Attention 的機制(下面會細講)，把 encoder 的 output 經過一些處理傳給 decoder。

（二）Attention mechanism(2%)

How do you implement attention mechanism? (1%)

在一開始純粹只有 encoder-decoder 的 model 中，我做出最好的結果為：

0.64/0.277 以及一些 output 例子：

```
Men are racing on a track. eos
a soccer player in a goal

A lion is walking in a pen. eos
a  is  into a

A man is playing a violin on the roof. eos
a man is walking in a

A person is adding garlic pieces in the pan. eos
a woman is adding a liquid into a pan
```

加上 Attention

下一題當中我會介紹我選擇出此 attention model 的方法，在此我先介紹如何實踐。

我把 encoder LSTM output 與它自己經過 softmax 之後的權重做結合，在一起給 decoder 拿去 train。

有意思的是，我的實驗結果是將 concatenate 結合之後的 attention 直接跟 **decoder LSTM output** 接在一起(concat)，然後直接接上 dense → 輸出的效果，會比把 attention 丟進 decoder LSTM 當一部分 input，效果更加的好(詳細說明會在後面提到)。

```
# Imporvising
# output 線性轉換結果為permute 2
permute1 = Permute((2,1))(encoder_outputs)
dense1 = Dense(39)(permute1)
permute2 = Permute((2,1))(dense1)
dense2 = Dense(latent_dim,activation='softmax')(permute2)

# Use Concatenate to merge dense2 and permute2,
# and then transfrom to (output_seq_len, latent_dim)
attention = concatenate(inputs=[dense2, permute2])
attention = Dense(latent_dim)(attention)
```

```
# Merge attention and lstmoutput
merge_att_lstm = concatenate(inputs=[attention, decoder_lstm_outputs])
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(merge_att_lstm)
```

分數約為 0.66/0.28 相比其他方法分數並不是特別高，但是結果較為理想。

```
Men are racing on a track. eos
a group of men are playing and the team

A lion is walking in a pen. eos
a dog is running in a field

A man is playing a violin on the roof. eos
a man is riding a

A person is adding garlic pieces in the pan. eos
someone is oil in a bowl
```

（三）How to improve your performance (1%)

(Describe the model or technique (0.5%) Why do you use it (0.5%))

關於 Attention Model，我另外嘗試了幾種 Attention 作法：

我上網參考了一個現成的 Attention Layer 並加以更改(使用 dot operation)，他所吃的 input shape 是 (sample_size, timestamps, features)，output shape 為(sample_size, features)利用這個 Layer 我可以把我的 encoder LSTM output 轉成一個(sample, latent dimension)的 tensor。照理說，這樣的作法已經把每個時間點的特性壓縮在不同的 cell 裡面，因此我只要讓我的 decoder 有用到這些 tensor 他就會自己學會怎麼考慮 time-stamp。

於是我嘗試了如下幾種作法：

a. 直接把這個 tensor 當作 initial state 傳給 decoder LSTM: 參數相同情況下分數略有進步下圖左(0.64, 0.28)，也可以透過參數調整找出比較高分的句子(下圖右,分數為 0.69, 0.29)，相較沒有做的情況下分數跟結果都有進步，辨識出許多單字。

```
Men are racing on a track. eos
two men are playing in a race

A lion is walking in a pen. eos
a dog is running in a woods

A man is playing a violin on the roof. eos
a man is riding a bicycle

A person is adding garlic pieces in the pan.
a woman is cooking a skillet
```

```
Men are racing on a track. eos
a soccer is playing in a and playing the

A lion is walking in a pen. eos
a monkey is running a grass

A man is playing a violin on the roof. eos
a man is doing

A person is adding garlic pieces in the pan. eos
a person is adding oil
```

b. 把這個 attention layer 的 output 透過 **Repeat vector** 複製 output_seq_len 次，然後跟 decoder input 合在一起，丟進去 train。結果可以衝到**很高的分數**，達到 **0.74/0.285** 但是分數高的時候往往預測出來都是殘破的句子（下圖左）

```
Men are racing on a track. eos
a are in a

A lion is walking in a pen. eos
a man is a

A man is playing a violin on the roof. eos
a boy is a

A person is adding garlic pieces in the pan.
a woman is a into a
```

這兩個小技巧都可以幫助我分數上升，但考量到最後是要給人看，不是比分數，就沒有在最後的 model 裡面 implement。

（2）將 attention 接到 decoder lstm output 之後，有效增進最中文字的可讀性：

叫詳細的比較資訊在下一題，我覺得可能的原因在於，讓 dense 直接看 attention 效果比較好，透過 LSTM 可能稀釋了 attention 裡面的資訊。

（3）調整總字數：我去掉了所有指出現一次的字及大部分出現兩次的字，到最後的調整結果，覺得在 2300 字左右表現不錯。進步幅度約為 0.3 分(0.22 → 0.25)

（4）裁剪 max seq len：因為我發現我最終預測出來的句子都不長，所以就直接讓 caption 讀進來的時候，就限制他的長度，比最高再短一點也沒關係。這樣避免 dimension 太高可以增進最後面的運算效率。

（四） Experimental results and settings (1%)

MODEL 方面

（1）使用 concatenate attention model vs multiply attention model

使用 multiply (0.66/ 0.279)

```
Men are racing on a track. eos
a soccer player playing a goal

A lion is walking in a pen. eos
a man is running in a field

A man is playing a violin on the roof. eos
a man is walking on a

A person is adding garlic pieces in the pan. eos
a person is pouring a liquid into a pot
```

使用 concatenate (0.68, 0.278)

```
Men are racing on a track. eos
a group of men are playing and the team

A lion is walking in a pen. eos
a dog is running in a field

A man is playing a violin on the roof. eos
a man is riding a

A person is adding garlic pieces in the pan. eos
someone is oil in a bowl
```

（2）將 attention 接在不同 decoder LSTM input vs decode LSTM output

接 input （ 0.67, 0.27）

```
Men are racing on a track. eos
a man scores a goal in a field

A lion is walking in a pen. eos
a dog is running in a field

A man is playing a violin on the roof. eos
a man  a

A person is adding garlic pieces in the pan. eos
a man is adding oil on a skillet of diced vegetables
```

接 output (同上圖)

```
Men are racing on a track. eos
a group of men are playing and the team

A lion is walking in a pen. eos
a dog is running in a field

A man is playing a violin on the roof. eos
a man is riding a

A person is adding garlic pieces in the pan. eos
someone is oil in a bowl
```

（3） Learning rate 與 Epoch ：Learning Rate 越低，epoch 越大，train 出來分數越高，越精準，但句子越難理解（殘破不堪）：下圖左：lr=0.0003，分數(0.745/ 0.27)

```
Men are racing on a track. eos
a are in a

A lion is walking in a pen. eos
a man is a

A man is playing a violin on the roof. eos
a boy is a

A person is adding garlic pieces in the pan. eos
a woman is a into a
```

```
Men are racing on a track. eos
a group of a soccer ball in a ball

A lion is walking in a pen. eos
a man is riding a bicycle

A man is playing a violin on the roof. eos
a man is running

A person is adding garlic pieces in the pan. eos
a woman is pouring a shrimp into a skillet
```

（4） 不同曾的 activation function : adam（上圖右）實驗結果為收斂較慢，最終句子還可以，分數低(0.63/ 0.26)