R06946003 資料科學碩一 湯忠憲 **Video Captioning**

Model description

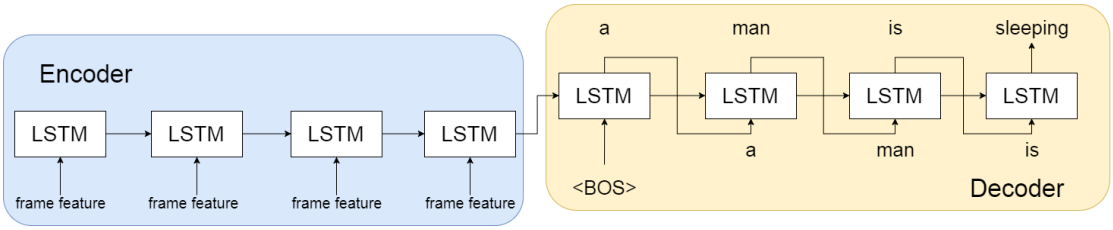

Figure1. Seq2Seq model

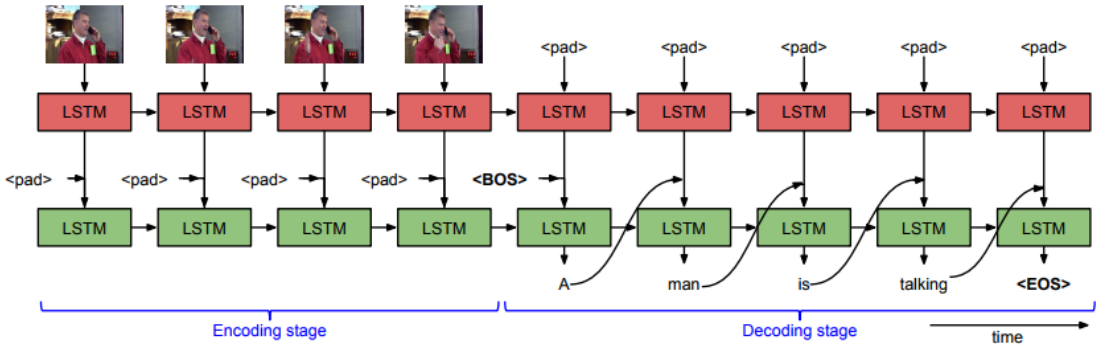

Figure2. Stacked LSTM Seq2Seq model[1]

本次作業主要是使用 Seq2Seq 為概念的 model，而作業 PPT 中有提到兩種架構，一種是分別用兩個 LSTM 來 model encoder 和 decoder 的 Seq2Seq，另一種是 Video to Text 中所使用的 stack structure。而後者的架構中並沒有明確的 encoder 跟 decoder，取而代之的是用 time step 來區隔的 encoding stage 和 decoding stage，並且形成共用參數的 LSTM，能夠降低模型複雜度。本次作業我也實作了兩款模型，可以發現 Stacked LSTM 的會比較快產生出正常通順的句子。因此以下的實驗也以第二種模型為考量。

Best Model detail:

每一個 video 約為 5 到 20 秒不等，透過 CNN 抽取 4096 維的 frame feature，共 80 frames/video。詞彙(token)則是全部都用上，共 6483 個詞彙。並且希望產生的句子長度在 15 個字以內。總結上述，架構參數的部分主要:

- input dimension = 4096;
- LSTM hidden dimension = 256;
- video LSTM step = 80;
- caption LSTM step = 15;
- 使用 ADAM optimizer，learning rate = 0.001

Training:

- Batch size = 32

[1] http://www.cs.utexas.edu/users/ml/papers/venugopalan.iccv15.pdf

Attention mechanism

How do you implement attention mechanism?

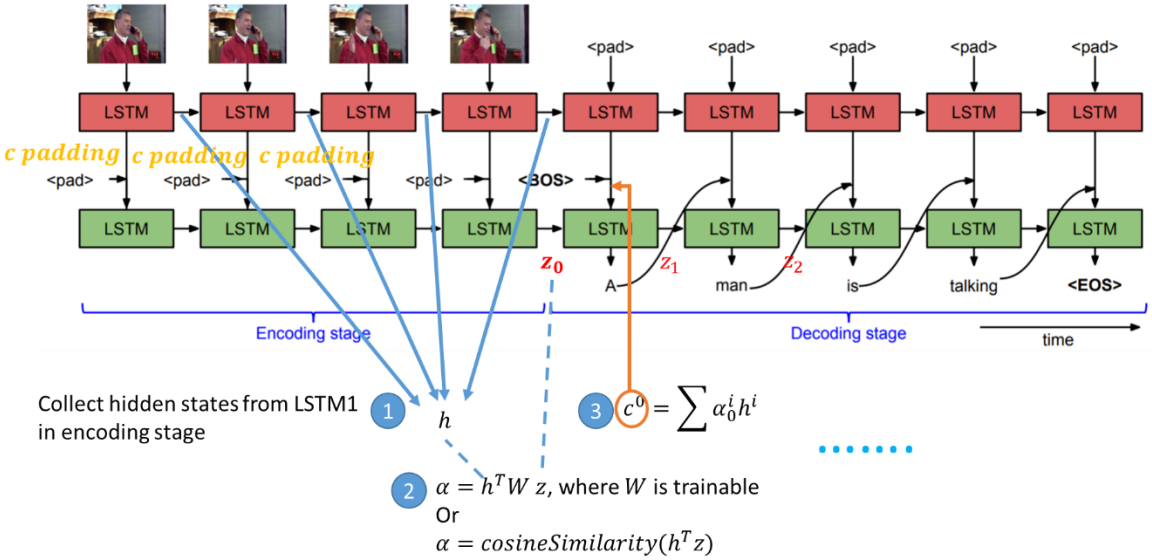

Figure3. Attention mechanism

依照課程 PPT 講解之 attention-based model 實作:

初始化: $z_0 = randomUniform; W = randomUniform; c\ padding$

1. 於 encoding stage 時紀錄 LSTM1 的 hidden state。
2. 透過 matching function ($h^T W z$) 求得α。
3. 將 α 與 h"們" 做 weighted sum，得到 c，並把 c 給到 LSTM2 作為其中一個 input。
4. 以新的 hidden state 2 更新z

重複以上操作。(在 encoding stage 時，用$c\ padding$作為 LSTM2 的其中一個 input。)

Compare and analyze the results of models with and without attention mechanism.

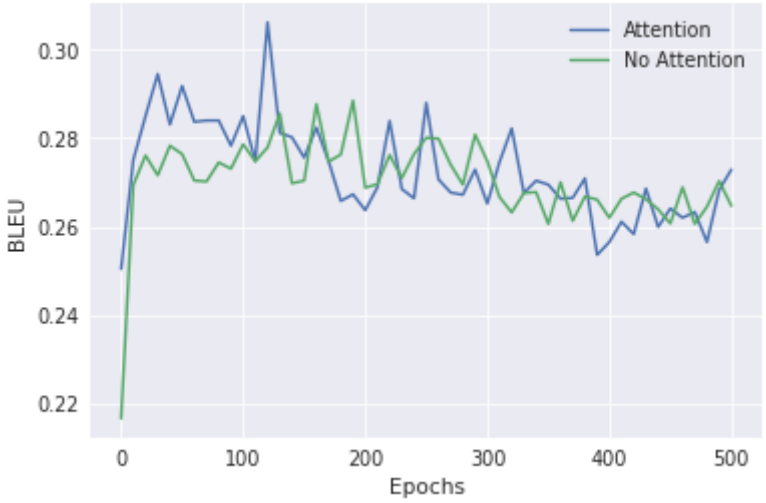

Figure4. BLEU score comparison (w/o attention)

在 Figure4 比較了有使用 attention 跟沒有 attention 的 model 在 validation set 上的表現。可以發現在前幾個 epochs 上，attention-based model 有較好的 BLEU score，而到了越後面兩者的 BLEU 均下降。應該是因為模型已經開始 overfitting，使得兩者在 validation set 上的表現下滑。

How to improve your performance?

Write down the method that makes you outstanding

1. Attention mechanism:
   如上提的觀察，使用 attention 可以在前幾個 epochs 有不錯的表現。此外，不同的 matching function 可以有不同的效果。其中，dot product 的方法可以 train 的較快，cosine similarity 在 build graph 時就比較久，可能是因為相較於單純 dot product 需要做 magnitude 的額外運算。

2. Sample technique:
   由於每一個影片都會對應到十幾句的 captions，共約 28000 筆 captions。若將跑一次 28000 筆 captions 當一個 epoch 的話，BLEU 大約為 0.223。不過若是每個 epoch 都從 28000 筆抽 1450 筆 captions (即 video 數量) 的話，BLEU 可以較穩定的維持在 0.25 以上。可能的原因我想應該是前者的 shuffle 頻率較後者低。

3. Curriculum learning[2]:
   在圖像辨識中，可以讓機器先學簡單的幾何圖形並漸漸增加圖像複雜度，這樣能神經網路訓練的速度更快。在本次作業，我嘗試將句字的長短做排序，當成複雜度的排比，並從較短的 caption 開始學。最後結果並沒有看到顯著的收斂速度的提升。可能原因我想有兩個: 1)這次的資料量不算多，因此本來的做法就能很快的收斂。在 validation set 我們可發現約 50 個 epochs 就可以有不錯的效果了。2)經排序的 captions 在某種程度上缺乏隨機性，與 shuffle 的效果打平了 (神經網絡從意料之外的樣本中學習最快)。

4. Scheduling sampling[3]:
   為能模擬真實 testing 的情況，我也嘗試使用 scheduling sample 的方式來 model 產生 input 的方式。Paper 中主要提到三種方法，我挑選其中的 linear decay 跟 Inverse sigmoid decay。不過最後效果並沒有明顯的差異，可能 converge 的速度需要再調整。

[2] https://ronan.collobert.com/pub/matos/2009_curriculum_icml.pdf
[3] https://arxiv.org/pdf/1506.03099.pdf

Experimental results and settings

Settings

綜合上述技巧，我作了有關 attention 和 scheduling sampling 的實驗，並觀察在不同 hidden dimension 下的變化。實驗設定如下:

A. 不使用 attention，hidden dimension 為 64 維
B. 不使用 attention，hidden dimension 為 256 維
C. 不使用 attention，hidden dimension 為 1000 維
D. 使用 attention，用 Inverse sigmoid decay 作 scheduling sampling，hidden dimension 為 256 維
E. 使用 attention，用 linear decay 作 scheduling sampling，hidden dimension 為 256 維

Results

從 Figure5 中可以發現不管什麼方法，在超過 100 個 epochs 後表現都會下降。在實驗上述實驗設定中，A,B,C 的結果可以發現 hidden size 設成 256 是一個不錯的選擇。實際去看產生的句子可以發現 hidden size 較大的 generator 產生的句子較長，例如給定相同影片: 256 dims) a woman is walking，1000 dims) a man and a deer walk through the woods。我覺得 1000 dims 因為產生更多字使得 BLUE 分母太大，降低整體得分。256 dims 則是產生得比較保守一點。

D 與 E 的實驗可以看出兩者在前幾個 epochs 差不多但是 E (Linear)在約 50 個 epoch 後有一波下跌，因為到了那裡 true previous token probability 大約為 0.6。不過幾個 epochs 後表現又上來，應該是因為隨機抽樣，使得就算上次拿到的是 predictive token，多抽幾次後也有可能抽到 true token。相較之下 D (Inverse sigmoid)則表現得比較穩定一點，我想應該是在遇到那個斜率較大的 drop 前，機器就學的差不多了，因此沒有很大的表現差異。

Figure5. BLEU score comparison (hidden dim, decay method)