MLDS HW 2-1 Video Caption Generation

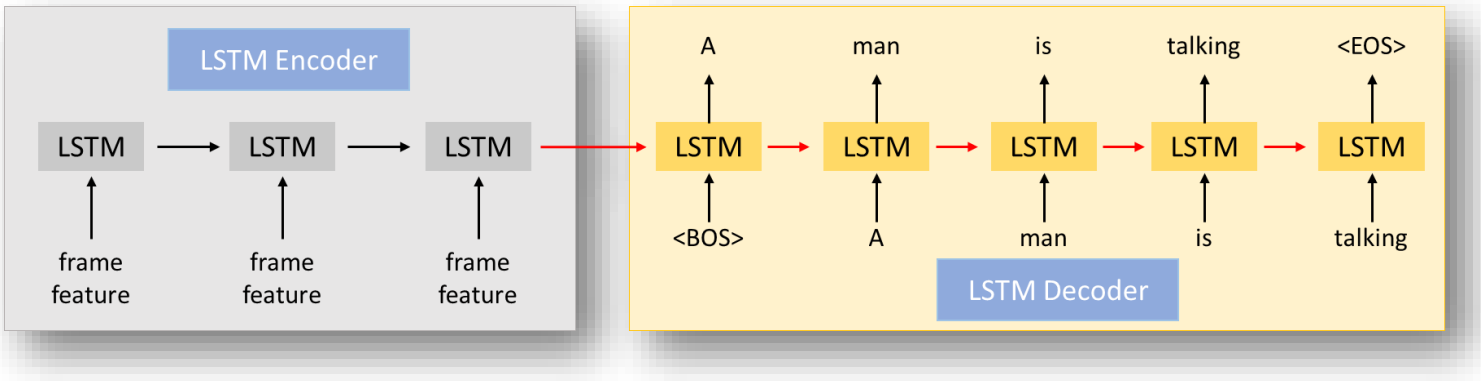
A. Model description:

Data Preprocess:

a. Build word dictionary (including word2ix, ix2word and bias_init_vector) :
首先從 training data 及 testing data 的 caption 中收集詞彙， 共有 6347 個詞彙，並依照所設的詞彙出現次數 threshold(2)，將總詞彙壓縮為 3870 個；根據所收集的詞彙建立 word2ix, ix2word 字典，以及 bias_init_vector 以提供模型訓練及預測使用。

b. Training data preprocess:
training data 中的每一個 video 長度約為 5~20s, 利用 CNN 對每個 video 抽取共 80 個 4096 維的 frame features. (助教提供)
將 training data 中的每個 caption 開頭加上<bos>，結尾加上<eos>，不足 16 字的 caption 做 padding (加<pad>) 至 16 字，並捨棄大於 16 字的 caption； 最後將每句 caption 中的每個詞彙根據字典 word2ix 轉成相對應的 ID vector。
由於一個 video 會有多個 caption，此 model 會對於每個 video 隨機選擇其中一個 caption 作為 label，對應相應的 frame features，生成訓練資料。

Model Structure



Model 架構為 encoder-decoder 架構，由兩層 size 為 256 維的 LSTM 所組成並共享權重；一個單層的 LSTM 作為 encoder，encode pretrained 的 CNN frame features，另一個 LSTM 作為 decoder，吃入 video embedding features 及前一次 decoder 的輸出作為 input，產生該 video 相對應的 caption。Loss function 使用 cross entropy，而 back propagation 則選用 AdamOptimizer 來更新模型的參數。

Inference:

將 input test video 中 80 個 frame 的 features 輸入 encoder 中，接著 decoder 讀到<bos>時便開始運作: 在 time step t 輸出經 softmax 後機率最大(argmax)的詞彙；並將此詞彙作為 time step t+1 的 decoder input ，持續重複此步驟直到 deocder 輸出<eos>後結束，最終產生此影片的 caption。

Model details

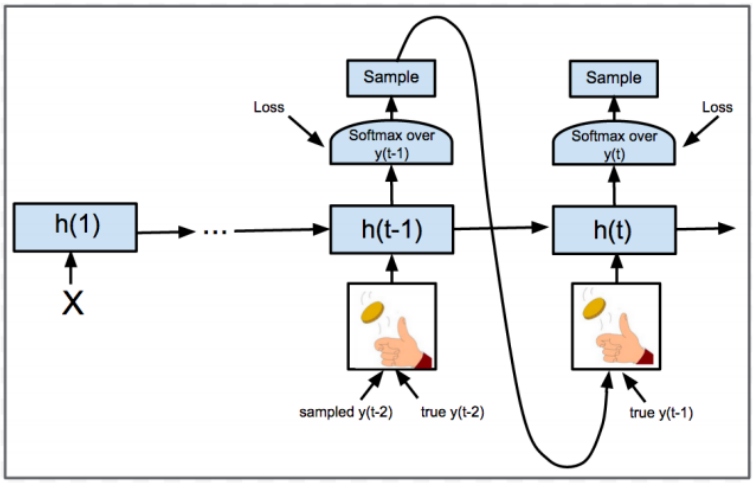
Parameters

Input dimension = 4096;
LSTM hidden dimension = 256;
#video LSTM steps = 80;
#caption LSTM steps = 16;
Optimizer = Adam optimizer
Learning rate = 0.001
Batch size = 32
Epoch = 205

B. How to improve your performance:

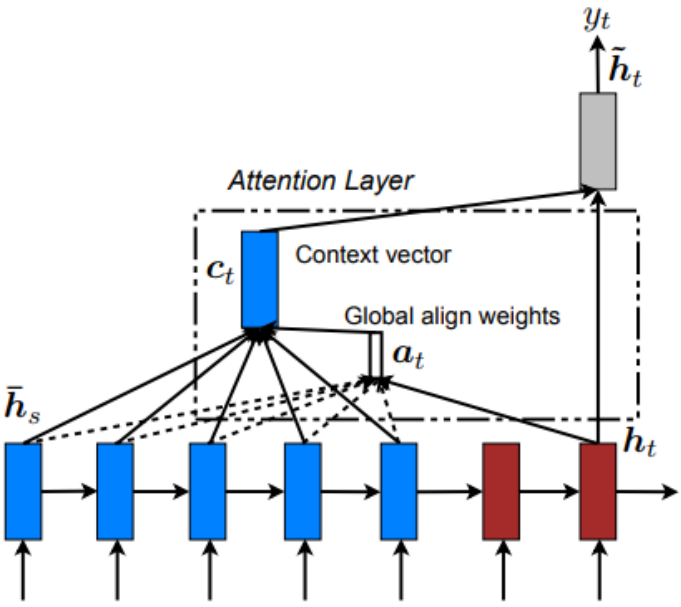
Write down the method that makes you outstanding

a. Schedule sampling



model 在 training 時把 ground truth 作為 RNN 的輸入(teacher forcing)，而在 inference 時則將 RNN 上時刻的輸出作為下時刻的輸入(last time step's output)；為了模擬 model 在 inference 的真實情況；可以在開始訓練時先以 ground-truth 來做輸入；隨著迭代次數的增加，model 的參數逐漸收斂，可以逐漸以上時刻的輸出作為下時刻的輸入；而具體的實現作法是以丟擲硬幣來決定。而這次的實作是利用 tensorflow 中的 ScheduledEmbeddingTrainingHelper 來實作，根據廣義的 bernoulli distribution 決定輸入的型態。

b. Attention mechanism



Attention mechanism 的實作是採用 tensorflow 中的 LoungAttention layer，將所有的 encoder output 作為 memory 傳進 attention layer, 每個 decoder output 會經由 context base function 對每一個 encoder output 計算 alignment score； Normalize 所有 scores 後，將每個 encoder output 與相對的 score 相乘後加總(weighted sum)以得到 context vector. 最後同時考慮 context vector 與 decoder output 以得到最後的結果。

Why do you use it

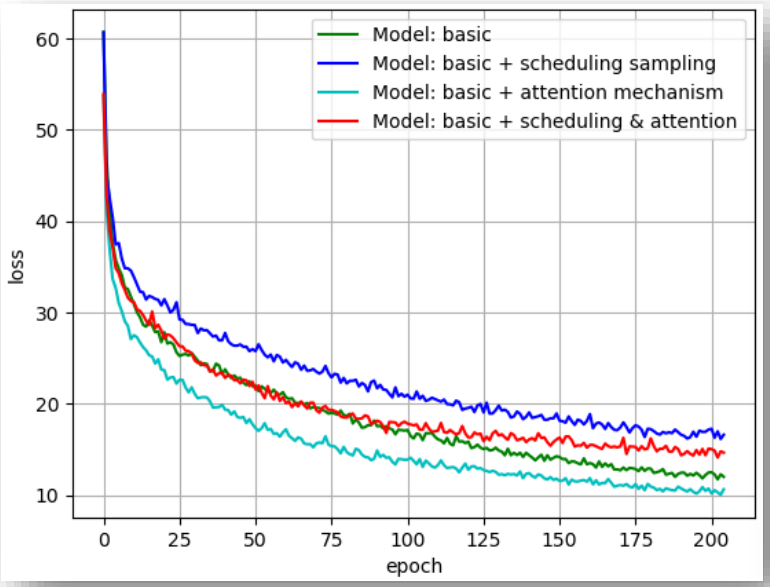
a. Schedule sampling

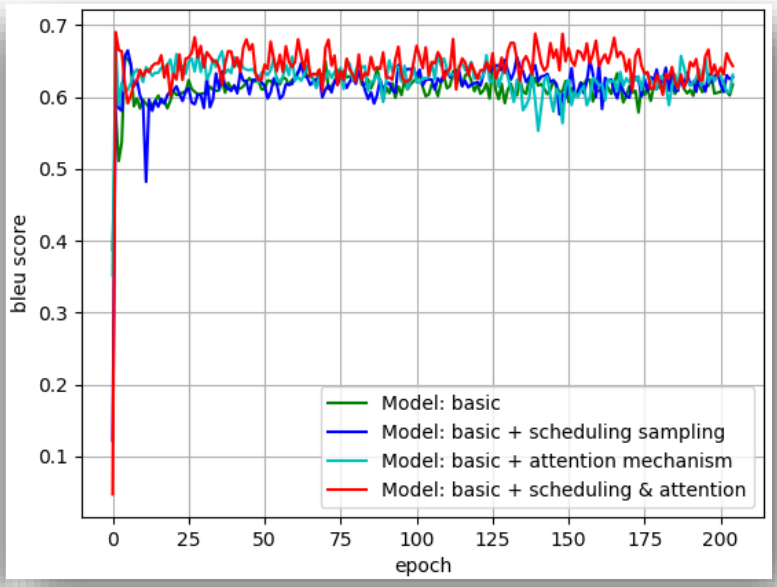
為解決 exposure bias 的問題，即 RNN model 在 training 時接受 ground truth input，而在 inference 時卻是接受 last time step's output，若在 t-1 時刻的到較糟糕的輸出結果，由於 model 無法得知真實的輸出，model 只能根據此糟糕的結果去預測下一時刻 t 的輸出，造成一步錯，步步錯的情形。

b. Attention mechanism

在 Encoder-Decoder 的結構中，輸入序列無論長短都會被 encode 成一固定長度的向量，decode 時則受限於該固定長度的向量；當輸入序列較長時難以保留全部的資訊，使 model 的性能變得很差。Attention mechanism 可以幫助 model 對輸入的每個部份賦予不同的權重，抽取出更加關鍵且重要的資訊，使模型能做出更加準確的判斷。

Analysis and compare your model without the method





根據上圖可知 model 在 epoch > 175 後 BLEU score 變化不大

a. Schedule sampling

從上圖觀察得知，在 model 加入 schedule sampling 後的 loss 上升，且 BLEU score 的差異不大；推測原因可能為 schedule sampling 需在 epoch 數大一點才能看出效果，training epoch 數太小反而會適得其反。

b. Attention mechanism

從上兩張圖觀察可知，在加入 attention mechanism 後確實能有效降低 loss，而在 BLEU score 的表現上，在前幾個 epochs 中 attention-based model 確實有較高的 BLEU score，而到訓練後期，basic model 和 attention-based model 在 validation set 上的表現雖差異不大，但仍可看見確實有改善。

c. Schedule sampling + Attention mechanism

從上圖中可看出，當 basic model 同時加入 schedule sampling 和 attention mechanism 後，雖然 loss 比只加入 attention mechanism 的 model 來得高(推測可能是受到 schedule sampling 的影響，在 epoch 數小時成效不佳)，但是 BLEU score 有明顯增加，大多數都在 0.6 以上；可見這兩個技巧的確對 model 的訓練確實有幫助。

C. Experimental results and settings

實驗中發現，epoch 在大於 175 後 BLEU score 就已變化不大(收斂)，再繼續 train 會發生 overfit 而導致 BLEU score 下降。最終使用以下的設定，以得到最佳的 BLEU score

Input dimension = 4096;
Word threshold = 2 (3870 words);
Luong Attention;
Schedule sampling rate = 0.2;
LSTM hidden dimension = 256;
#video LSTM steps = 80;
#caption LSTM steps = 16;
Optimizer = Adam optimizer;
Loss = sparse_softmax_cross_entropy_with_logits;
Learning rate = 0.001;
Batch size = 32;
Epoch = 205;
Max BLEU score = 0.68

D. 分工:
