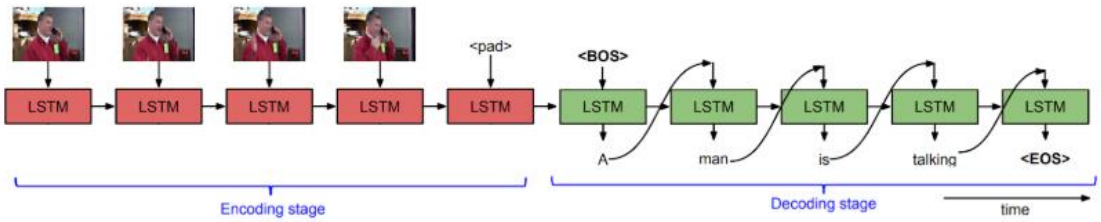
Model description

我們使用以下 sequence to sequence 結構:



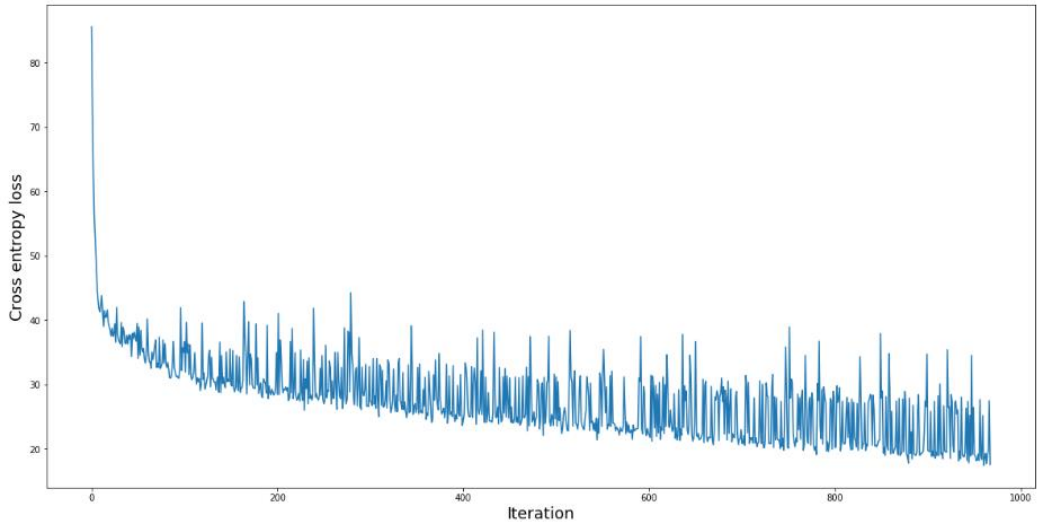
RNN cell 使用的是 GRU，而 decoding stage 有使用 attention。Encoder 會將 input video 的 4096 維先降到 512 維，然後將 512 維的 input 和 256 維的 hidden layer 丟進 GRU cell 中，取 encoder 最後的 hidden layer 作為 decoder 的初始 hidden layer，並將所有 timestamp 的 hidden layers 存起來作為 attention 用。Decoding stage 中，對於每個 timestamp，decoder 會根據當下的 input (一個單字的 embedding，512 維) 與當下 hidden layer (256 維)，經過一個 DNN 去算出每個 encoder hidden layer (80 個，共 80 timestamp，shape: (80, 256)) 的加權比重，加權平均後將結果與 input concat 起來(768 維)，再降為到 512 維作為 decoder GRU 的 input。

How to improve your performance

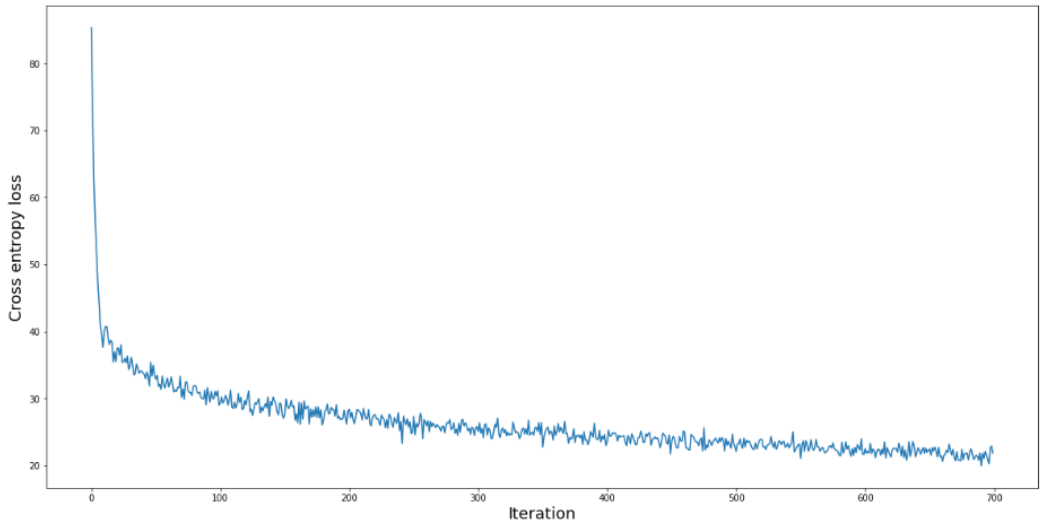
我們有使用 teacher forcing 和 attention。(實驗中的 performance 都沒有過 baseline，因為最後過 baseline 的 model 要 train 相當久的時間，以下的實驗數據每張圖大約 train 半小時。)

- **Schedule Sampling:** 原本沒有使用 schedule sampling，teacher forcing rate 為 1，可以較輕易的把 model train 下去，甚至 loss 可以降到 1 以下。但 test 時 output 就會出現和 train 一模一樣的句子。應該是因為純 teacher forcing 會使 model 被訓練成無論吃什麼 hidden layer(前面輸出過什麼句子)，只要 input token 是哪一個詞，output 就要是哪一個詞。因此不只無視影片實際內容，輸出與 training data 一模一樣的句子，甚至常輸出與影片根本毫無關係的內容(可能因為 hidden layer 被 model 無視了)，因此即使 schedule sampling 難 train 許多，還是要讓 model 學著自己輸出整句話。實驗結果當 train 大約 1000 次 iterations 後，有 schedule sampling 的雖然 loss 較高，但在 testing data 上的表現較好。

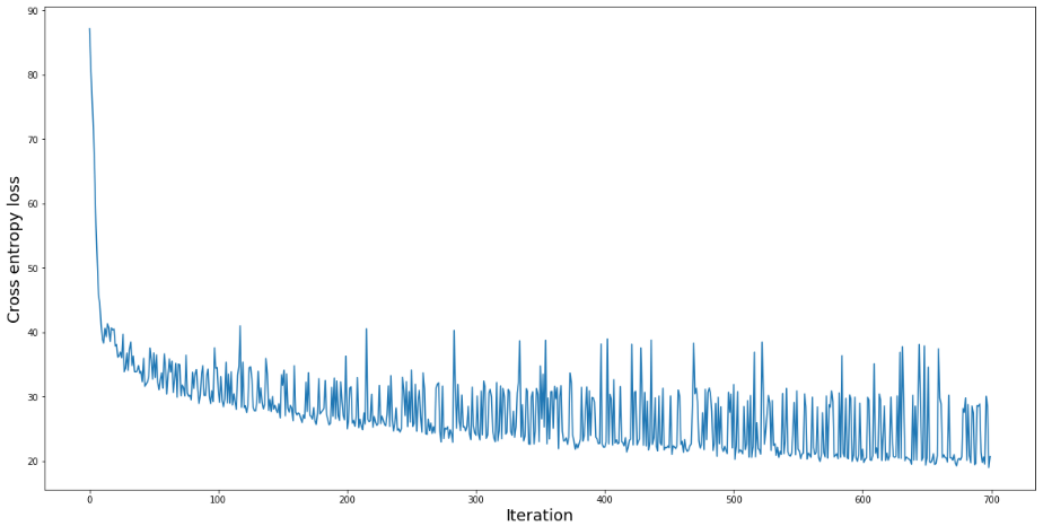
  teacher forcing rate 0.8: (testing average BLEU: **0.58326**)



teacher forcing rate 1: (testing average BLEU: 0.574403)



- Attention: 有沒有使用 attention 在 training loss 上看不出明顯差異，下圖是沒有 attention 的版本，與最上圖的 model 相比只有把 attention 部分拿掉，其他維持不變。

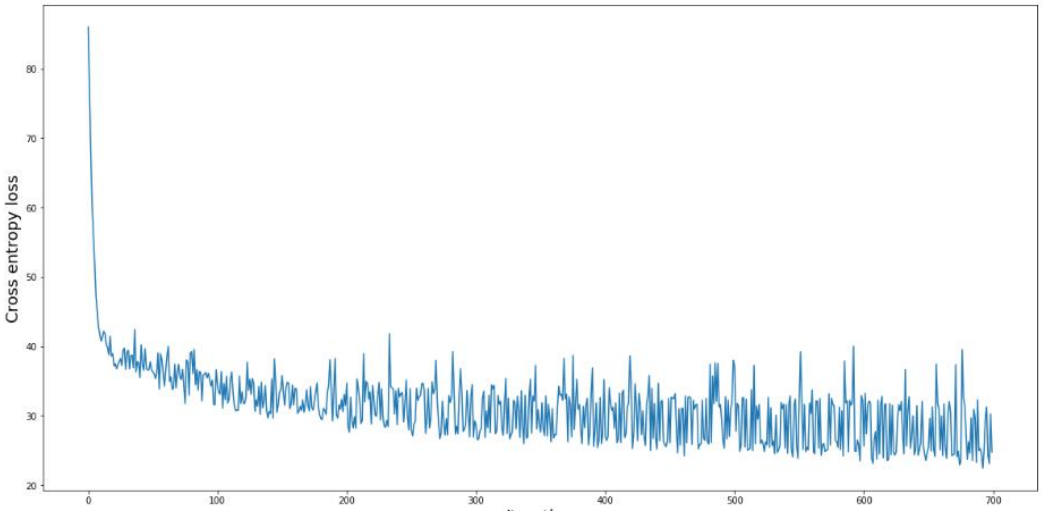


可以看到甚至收斂得比有 attention 的還要快。但這似乎只是因為 attention-

based model 的參數量較多較難 train。在 testing data 上的表現，沒有 attention 是遠差於有 attention 的，BLEU 僅 0.517349。同時也會有類似純 teacher forcing 的問題，就是 output sentence 與影片關聯性不大，就只是將 training data 的句子拿過來用。我想也是因為僅靠 encoder last hidden layer 來 decode 還是太吃力。

Experimental results and settings

- 原本在做 attention 時是將 encoder hidden layers 做 weighted sum 並和 input token embedding 做 concatenate 後直接餵進 GRU，但 train 的結果並不理想，同樣 1000 iterations，loss 會比後來的版本高出 2 到 3:



後來改了 model 架構，讓 input token embedding 做 concatenate 後先過一個降維 DNN 再餵給 GRU 後，效果就好了許多。

- 原本 RNN cell 的 hidden layer dim 調很高，至少 512，甚至 1024。但 train 的效果極差，常常根本 train 不下去。後來一次又一次調小，發現居然都還是能 fit training data，256 維甚至 128 維就還足夠。
- Image Captioning 中每個影片都有多個 label，要讓一個 model 同時 fit 這些所有 label 似乎很難，因此我原本只有 fit 固定一個 label。的確很容易 train 下去，甚至可以讓 cross entropy loss 低到零點幾，輸出和 training data 完全一模一樣的句子。但 testing data 上會炸掉，model 根本沒學到有用的資訊，只是硬記什麼 input data 要產生哪一句話，而沒有實際根據影片的內容。後來發現，將固定 label train 好的 model 拿來繼續 train 另一個 label，雖然一開始 loss 很高，但 train 到完全 fit 新 label 後，並不會使原本的 label 完全炸掉。所以後來為了更快速的收斂我會先 train 固定 label，然後換一個 label train，然後再換一個。最後再 random label。我覺得概念上有點類似一開始先 teacher forcing 強迫 model 學固定的東西，再漸漸讓它自由發揮。
- 原本我的 output length 設為 30，但後來發現將 length 改短成 10 或 15 時，train 出來的效果會好很多。我覺得是因為 training data 中的 label 大部分還

是都在長度 15 以內，當 output length 設為 30 時，等於要 PAD 將近一倍的長度。可能會因為 gradient vanishing 使得 model 很容易學到 EOS 後要接一堆 PAD，但開頭真正該講什麼話卻都沒學到。後來發現 bucketing 似乎也是類似的概念，因此將長度差太多的句子放到不同 batch，用不同的 output length 來 train。不過因為時間因素就沒有再實作 bucketing 了。

分工表:

t06902115，張晉之:

- 資料前處理: 建字典、建 dataset
- 前期 model 架構: 有加 attention，但還沒有 batched，train 起來速度較慢
- Train 在較小的 dataset 上測試 model 的可用性

b03902130，楊書文

- 後期 model 完成: 改寫 model 與 dataset 使其可以 batched 的多句同時 train
- Tune model: train 在完整 dataset 上，調整各 layer 參數與 model 架構
- Beam search
- Report