MLDS 第二次作業 Report
R05921012 吳宗澤

- Model description (2%)
  - Describe your seq2seq model

我主要是模仿助教所提供的 S2VT 模型去實作，首先對輸入的文字 caption 用 nltk 做 tokenize 的動作，然後用 sklearn 的 CountVectorizer 去做 Label 和塞選掉出現機率的低的字。並把影像的 features，<bos>，mask，Labels, batch_size 作為 input 的 placeholder 到 model。Model 主要也是有兩層 256 的 BasicLSTMCell Layers,Encode 時先把前 80 timesteps 輸入到第一層 lstm,並用第一層的 lstm output state 跟 padding 的 zeros 做 concat 放入第二層 lstm 的 input。然而 Decode 階段，就會在第一層 input padding 的部分並把 output 和 ground truth label 做 embedding 的 tensor 或上一次第二層的 output 做 concat 當成第二層的輸入（Schedule Sampling）。除此之外，因為 dataset 一個影片有個 caption 可以使用，我做了一個機制讓 loss 低於一個會跟著 training epoch 下降的 threshold 才去做新的隨機抽取 Label caption。這樣可以讓我的 Model 朝向最難讓 loss 下降的那些 captions 做更新，而最難的下降的那些照理來說就都是比較長和難學的 captions，讓他有一個 curriculum learning 的感覺。

- Attention mechanism(2%)
  - How do you implement attention mechanism? (1%)

我嘗試了兩種，第一種是將前 80 個 feature 在第二層的輸出的 tensor 給收集起來並和接下來第二層的 cell 的 hidden states 做 concat 然後丟進一個 NN 裡面並做 softmax，在把 softmax 的結果與前 80 個 feature 在第二層的輸出的 tensors 做內積與第一層的 output 和 schedule sampling 出來的 tensor 堆疊起來當做第二層的輸入，但是實作後發現運算量太大，會讓電腦 G P U 記憶體爆掉。因此也嘗試過將前 80 個 feature 在第二層的輸出的 tensors 與第二層要輸入的 hidden states 做 cosine similarity 再輸入 softmax layer 拿到機率去跟前 80 個 feature 在第二層的輸出的 tensors 做內積。利用 cosine similarity 的確可以將計算量減少很多不然預測一個字要做 80 次的 nn 輸入輸出，有點太誇張。

  - Compare and analyze the results of models with and without attention mechanism. (1%)

覺得做 attention 並沒有讓我的 model 進步太多不知道是否是因為因為資料量太少，讓他學不會該關注在哪個 frame 上的 features，或者是 cosine

similarity 不算一種好的評分方式，因為不一定 distribution 長得越像就要多 attent 在那裡，也許應該找一種更好的評分方式讓 attention 的這個機制更能學到我們想要的效果，曾經想要使用ＲＬ去學這塊，但是沒有足夠的時間去實作。

- How to improve your performance (1%)
    - Write down the method that makes you outstanding
    - Describe the model or technique (0.5%)
    - Why do you use it (0.5%)

我覺得使用 schedule sampling 的確可以讓我的 loss 下降到不錯的地方，稍微做個簡單的 filter 讓太少出現的字也的確讓我的效果有提升，也將那些太少的字歸為 unknown 讓 embedding 比較好學到字與字的關係，而且我也做了測試有無做 bos 和 eos 的差別，沒有加幾乎會讓效果盪到谷底，做此測試的原因是因為我原本想說當第一層的 input 消失變 padding 之後，第二層的 layer 應該會收到資訊說要準備換了產生文字囉，但沒想到作用沒有 bos 的作用高。另外我也測試了 AdamOptimizer 與 GradientDescentOptimizer 的差別，因為論文裡面很常使用ＧＤ或是ＳＧＤ來做更新，但我發現 Adam 效果竟然比ＧＤ好，且 loss 下降的速度也快很多。
其實在做 embedding 的時候我很猶豫到底要把 embedding 的 size 怎麼設會比較好，因為資料量很少，若用了過大的 embedding size 感覺容易 overfitting 讓測試時候的效果變差。但如果我不把出現次數少的 vocabulary 給刪掉，他又會容易找不到那些 words 之間的關係，因為次數少所以也學不到好的。

- Experimental results and settings (1%)
    - parameter tuning, schedule sampling etc

我在 training 的時候有把 training data 的 prediction 和 testing data 的 prediction 給 print 出來，然後和 ground truth 的 caption 做比較也去觀察 loss 的變化，因此沒有去讓 training epoch 結束再去做 bleu score 的衡量。因此沒有很多的數據可以貼出來比較。

但我有發現我把 Adam 的 Learning rate 條大一點到 0.01 可以收斂比較快，然後實作我自己的隨機選 caption 方式也比每次都隨機收斂的快和好，我也有去嘗試讓 schedule sampling 有一個 decading 的 rate 讓他到後面越來越要用自己的前一個 output 做預測。因為很難抓那個 decading rate 大小,畢竟每 predict 一個字，然後就要 decade 一次，所以我將 decade rate 設到 0.9999999，這樣大概可以在 500 個 epoch 之後都用自己來預測，效果有稍微

好一些。