深度學習

HW11

學號 : B103012002

姓名 : 林凡皓

為了保持報告完整性，本次作業報告接續上次作業報告(從十三、Attention LSTM開始為本次作業)。

1. Image Feature Extraction

在Image caption model中，首先重要的是encoder，此encoder會接收一張圖片作為輸入，並生成用於解碼的特徵。

我們使用小型的RegNetX-400MF作為骨幹網路，以減少訓練時間。

對於vanilla RNN和LSTM，我們使用average pooled features來解碼，而對於attention LSTM，我們使用learning attention weight來聚合空間特徵。

1. Word embedding

在深度學習系統中，我們通常使用向量來代表字母。單字中的每個字母都會與一個向量相關，這些向量將會與系統其他部分一起學習。

這邊實作WordEmbedding類別來將字母轉換為向量。

* 實現方法 :

初始化部分，先創建一個參數W\_embed，其shape = (vacab\_size, embed\_size)。使用torch.randn來初始化該矩陣，並將其除以來進行標準化。

接著實作forward方法，這個方法主要是根據輸入的字母，從W\_embed中找到其對應的向量，可以透過index operation來完成。

* 執行結果 :



Error非常低，代表說該類別功能正確。

1. Temporal Softmax Loss

在RNN語言模型中，我們在每個timestep產生出單字中每個字母的分數。由於我們知道各個timestep的ground-truth，因此我們在每個timestep採用cross entropy loss。我們將loss進行時間上的總和，並在minibatch上進行平均。

但是這邊有一個問題，就是由於我們是對minibatch進行操作，而不同的caption可能會有不同的長度，因此我們在每個caption的尾端加上’<NULL>’以便它們都具有相同長度。我們不希望這些’<NULL>’也加入loss的計算，因此我們需要一個額外的參數ignore\_index來告訴程式碼在計算loss時要忽略掉那些index。

實作temporal\_softmax\_loss來完成loss的計算。

* 實現方法 :

透過torch.nn.functional.cross\_entropy來實現。

torch.nn.functional.cross\_entropy可以傳入的參數有input、target、weight、size\_average、ignore\_index、reduce、reduction、label\_smoothing。這邊主要為用到input、target、ignore\_index、reduction。

Input的部分即為x，不過關於維度的部分，我們需要將timestep的維度T放到最後，這樣才能使用cross entropy計算loss，因此可以使用permute(0, 2, 1)將timestep維度與最後的維度做交換。

Target 即為y。

Ignore\_index即為呼叫函數時的輸入，主要用來在計算loss時忽略掉一些標籤。

Reduction的部分要設置維’sum’，因為我們希望將計算出來的loss進行時間上的總合，最後再對minibatch取平均。

* 執行結果 :

根據不同的情況去計算loss，結果如下

一張含有 文字, 字型, 印刷術, 設計 的圖片

自動產生的描述

這些樹直接與預期的數值接近，代表說函數功能正確。

1. Captioning Model

我們要將所有東西封裝成一個captioning模組，該模組有一個通用的結構，可以根據cell\_type參數來控制要用於RNN、LSTM或是attention LSTM。目前只需要實作cell\_type = ‘rnn’的部分。

* \_\_init\_\_ :

1. 實現方法 :

初始化部分主要是要初始化output projection、feature projection、word\_embedding和backbone。

Output projection的部分為將RNN的hidden state映射到字母機率的層，可以透過Linear layer來做維度的改變。

Feature projection的部分為從CNN pooled feature映射到h0的部分，一樣可以利用Linear layer來做維度的轉換。

Word\_embedding的部分即為先前定義好的WordEmbedding類別。

骨幹網路的部分，可以透過創建一個已經定義好的RNN類別來實現。

* forward :

1. 實現方法 :

實作RNN forward path的部分來計算loss，backward path會利用autograd來實現。

首先要將輸入的字串分割成captions\_in和captions\_out。captions\_in為整個字串除了最後一個字母，而captions\_out為整個字串除了第一個字母。captions\_in即為RNN之input，而captions\_out為RNN預期的輸出。

接著要將輸入的字母做embedding，此部分利用word\_embedding實現。

關於輸入圖片，我們需要經過Encoder將圖片轉換為特徵，此部分透過小型RegNet-X 400MF來實現。

將透過Encoder得到的特徵x送進feature projection來映射成h0，並透過骨幹網路(RNN)來產生hidden state vector。

最後再透過output projection將RNN的hidden state映射到字母機率，並利用先前定義好的temporal\_softmax\_loss來計算loss。

1. 執行結果 :

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

由結果可以看到，計算出來的loss與預期的loss完全相同，代表說此函數功能正確。

1. Overfit Small Data

為了確認剛才實作的每一樣東西都可以正常運行，我們拿50張圖片來讓模型overfit。訓練結果如下

一張含有 行, 螢幕擷取畫面, 繪圖, 圖表 的圖片

自動產生的描述

可以看到經過80次的迭代後，loss來到趨近於0的程度，即成功的overfit資料。

1. Inference : Sampling Captions

Image captioning模型在訓練與測試階段和分類器的行為模式不同。

在訓練階段，我們將ground-truth的caption在每個timestep餵給RNN。

在測試階段，我們從單字的分布中採樣，並將樣本在下一個timestep中作為輸入餵給RNN。

實作CaptioningRNN.sample，並訓練模型以及對training data和validation data做採樣。

* 實現方法 :

在每個timestep中，我們會對當前的字母做embedding，接著將它與先前的hidden state送進RNN已取得下一個hidden state。利用此hidden state來取的每一個字母的分數，並將擁有最高分數的字母做為下一個字母。

首先將圖片經過encoder以取得特徵，接著利用feature projection將特徵映射成hidden state。

在每一個timestep中，透過先前定義好的word\_embedding來將字母做embedding，並將embed完的結果與前一個hidden state利用step\_forward來產生下一個hidden state。將hidden state利用output projection映射到分數，並從中取最高分做為下一個字母，將此字母存放到captions中。

為了確保每個樣本都以’<START>’作為開頭，因此要創建一個形狀為(captions.shape[0], 1)的張量，並將其與captions連接起來。

* 訓練過程 :

一張含有 繪圖, 行, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

由結果可以看出，訓練60個epochs後，loss會降到趨近於零的程度。

* 採樣結果 :

接著分別查看對training data與validation data做採樣的結果。

1. Training data :

部分結果如下

一張含有 哺乳動物, 柵欄, 戶外, 草 的圖片

自動產生的描述



一張含有 服裝, 人員, 男人, 復古風格 的圖片

自動產生的描述

由結果可以看出，對於訓練資料來說，RNN可以很成功的採樣出圖片的內容。

1. Validation data :

部分結果如下

一張含有 柵欄, 戶外, 樹狀, 人員 的圖片

自動產生的描述

一張含有 文字, 服裝, 幼兒, 男孩 的圖片

自動產生的描述

一張含有 輪胎, 車輛, 輪, 文字 的圖片

自動產生的描述

由部份結果可以看到，對於validation data來說，RNN基本上是無法順利採樣出圖片中的內容。

1. LSTMs : Step Forward

LSTM為改良版的RNN。一般RNN在處理長序列時會難以訓練，這是因為重複的矩陣乘法導致梯度消失或是梯度爆炸的問題。

LSTM透過在RNN的update rule中引入gate的概念來解決這樣的問題。

與RNN相同，在每個timestep先去計算activation vector 。接著將a分成四個向量 ，並透過sigmoid()、sigmoid()、sigmoid()分別取得input gate、forget gate、output gate，以及透過tanh()取得block gate。

最後透過以及來計算下一個cell state與hidden state (\*為elementwise product)。

Computational graph如下

一張含有 圖表, 行, 文字, 方案 的圖片

自動產生的描述

接著根據以上說明完成單一timestep的forward path，LSTM.step\_forward( )。執行結果如下



可以看到下一個hidden state與cell state的誤差很小，代表函數功能正確。

1. LSTM : Forward

完成LSTM.forward( )來對整個時間序列資料執行forward propagation。

* 實現方法 :

整個時間序列一共有T個timestep，利用for loop迭代T次，每一次迭代都執行一次剛才定義的LSTM.step\_forward( )。將每個timestep所得到的hidden state stack在一起，得到最終hidden state output。

* 執行結果 :



Hidden state output誤差很小，代表函數功能正確。

1. LSTM Captioning Model

修改CaptioningRNN類別，將其加入self.cell\_type = ‘lstm’的功能。

* 實現方法 :

這部分需要修改初始化與forward方法。

初始化的部分，output projection、feature projection和encoder接不需要修改，要修改的只有骨幹網路的部分，當self.cell\_type = ‘lstm’時，骨幹網路使用LSTM。

Forward path的部分，由於骨幹網路已經修改為LSTM，因此forward函數也會跟著修改成LSTM.forward( )。

* 執行結果 :

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

可以看到loss的誤差很小，代表說此模型可以正常運作。

1. Overfit Small Data

使用與在RNN時相同的資料集來讓模型overfit，執行80個epochs後結果如下

一張含有 繪圖, 行, 圖表, 文字 的圖片

自動產生的描述

最終的loss大約落在3左右。

1. Caption Sampling

修改CaptioningRNN.sample使得他也可以執行LSTM。

* 實現方法 :

利用if-else來判斷要執行RNN還是LSTM。

在LSTM的情況下，初始化除了要初始hidden state之外，還有cell state要做初始化。接著利用for loop與呼叫骨幹網路的step\_forward來取得下一個timestep的hidden state與cell state。

利用output projection來將hidden state映射到分數，並取分數最高的字母作為該timestep的最終結果。

最後再將每個timestep所得到的字母合併成最終結果。

* 執行結果 :

一張含有 繪圖, 行, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

在訓練60個epochs之後，loss降到10左右。

1. Test-time Sampling

查看訓練完成的模型對training data與validation data的效果。

* Training data :

一張含有 人員, 人的臉孔, 食物, 服裝 的圖片

自動產生的描述

一張含有 人員, 螢幕擷取畫面, 人的臉孔, 男人 的圖片

自動產生的描述

由部份結果可以看到，訓練60個epochs所得到的模型在training data上的採樣效果不太好。

* Validation data :

一張含有 服裝, 樹狀, 人員, 戶外 的圖片

自動產生的描述

一張含有 哺乳動物, 戶外, 雲, 草 的圖片

自動產生的描述

對於validation data也一樣，效果不太好。

1. Attention LSTM

在attention LSTM中會加入一個額外的輸入 ，與前一個hidden state一起輸入到LSTM中。

我們可以透過scaled dot-product attention來取得額外輸入。首先，我們先將CNN取得的特徵從映射到，得到映射後的activation A，並將A與前一個time step的hidden state用來產生attention weights，公式為，為A在time step t之attention weights。

為了簡化計算，我們對A和做flatten，得到以及。接著將送進softmax中，將attention weights作正規化。

給定attention weights，其attention embedding為。

根據以上說明，完成dot\_product\_attention的實作。

* 實現方法 :

首先要將A做flatten，原先A的shape為(N, H, 4, 4)可以透過reshape將A flatten為shape (N, H, 16)。接著透過公式計算attention weights，並將其送進softmax作正規化。透過view對做flatten，並利用公式取得attention embedding output。

* 執行結果 :



結果attention embedding output與attention weights的誤差都很小，代表說函數功能正確。

1. Attention LSTM : step forward

Attention LSTM forward path與LSTM forward path很相似，差別在於attention LSTM forward path多了attention input以及embedding weight matrix作為輸入。因此，activation vector的公式要修正為。

* 實現方式 :

透過activation vector的公式取得activation vector a，接著將a分成四個向量 ，並透過sigmoid()、sigmoid()、sigmoid()分別取得input gate、forget gate、output gate，以及透過tanh()取得block gate。最後透過以及來計算下一個cell state與hidden state。

* 執行結果 :



結果next hidden state與next cell state的誤差都很小，代表說函數功能正確。

1. Attention LSTM : forward

剛才實作一個time step的forward path，現在將它延伸到整個timeseries。

* 實現方法 :

利用for loop迭代total time step T次，每一次迭代都先透過dot\_product\_attention取得embedding attention weights，並將此embedding attention weights與attention input x、前一個time step的hidden state、前一個time step cell state送進activation vector的公式中以得到activation vector。最後將得到的hidden state output回傳即可。

* 額外說明 :

Hidden state output的shape為(N, T, H)，其中N代表batch size、T代表total time step、H代表hidden state size。在forward中計算的hidden state shape為(N, 1, H)，因為for loop為針對單一time step做計算。

* 執行結果 :



結果hidden state的誤差都很小，代表說函數功能正確。

1. Attention LSTM captioning model

接著修改上一次作業中定義的CaptioningRNN.\_\_init\_\_和CaptioningRNN.forward，讓此類別可以建立並訓練attention LSTM。

* CaptioningRNN.\_\_init\_\_ :

透過if else判斷現在要創建’rnn’、'lstm'或是’attn’模型，這邊只針對’attn’模型部分做解釋。

初始化部分主要是初始化feature projection與骨幹網路。

Feature projection為一個linear layer，input channel size與output channel size分別為input dimension與hidden state size。

骨幹網路為方才完成定義的AttentionLSTM class。

* CaptioningRNN.forward :

與RNN還有LSTM不同的地方在於，做feature projection時我們需要對input資料做permute以方便做projection。原先資料shape為(N, H, 4, 4)，我們需要將它轉變為(N, 4, 4, H)。映射完成後，再將shape轉回原來的樣子。其餘部分皆與上次作業中RNN與LSTM的部分相同。

* 執行結果 :

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Loss之誤差很小，代表說此forward path可以正常運行。

1. Overfit small data

透過遇上次訓練RNN的小型資料做訓練，看模型是否能夠overfit。

訓練過程如下

一張含有 行, 繪圖, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

一張含有 文字, 字型, 螢幕擷取畫面, 印刷術 的圖片

自動產生的描述

可以看到最後幾個epochs訓練loss來到5.5左右，表示模型可以順利擬和小資料。

1. Caption sampling

修改CaptioningRNN.smaple的部分來訓練attention LSTM。

* 實現方法 :

初始化部分要初始化A、hidden state、cell state。

A為feature projection的結果，根據CaptioningRNN.forward部分說明的方式做初始化。

Hidden state與cell state部分根據註解NOTE部分做初始化，這邊都初始化為A.mean(dim=(2, 3))。

在forward path的部分，先透過dot\_product\_attention取得attention embedding output與attention weights，再透過剛才定義的step\_forward來更新hidden state與cell state。最後利用output projection將hidden state轉換為分數，並取最大分數作為預測的單字。

* 執行結果 :

對整個訓練集做訓練，訓練過程如下

一張含有 繪圖, 行, 圖表, 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 文字, 字型, 螢幕擷取畫面, 印刷術 的圖片

自動產生的描述

可以看到最後loss為0.12，代表說模型overfit。

接著對部分訓練資料與驗證資料察看結果

1. 訓練資料 :

一張含有 食物, 盤子, 餐具, 資料表 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 速食, 食物 的圖片

自動產生的描述

一張含有 文字, 車輛, 公車, 陸上交通工具 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

可以看到對於訓練資料來說，目前的模型可以有不錯的預測能力。

1. 驗證資料 :

一張含有 室內, 牆, 浴室, 室內設計 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 文字, 小工具, 通訊設備, 電子裝置 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

對於彥徵資料來說，雖然模型沒辦法完美地預測出結果，但是可以看到模型基本上可以對圖片做出一些合理的解釋，表現比上次作業中的RNN以及LSTM好上許多。

1. 額外嘗試

* Attention-based Model :

Attention based model是模擬人類working memory的功用。

人類記憶體機制如下圖

一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

Attention based model可以分成兩類，分別為working memory和sensory memory溝通，以及working memory和long-term memory溝通的部分。

1. 第一類attention based model :

一張含有 文字, 螢幕擷取畫面, 字型, 圖表 的圖片

自動產生的描述

第一類attention based model的好處在於，假設輸入一個很長的序列，可能很難把所有資訊量一次處理好，attention-based model可以一次只注意input object的某個部分。

1. 第二類attention based model :

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

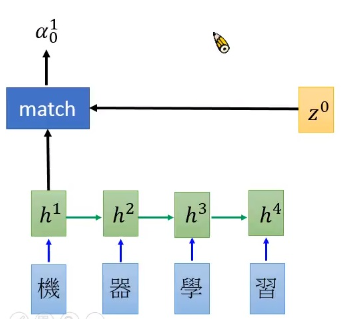
上圖為第二類attention based model，模擬working memory與long-term memory的溝通。

RNN / LSTM的問題在於memory沒辦法太大，因為memory越大，參數就越多。

RNN裡面，memory到memory之間有一個transition weight。如果memory size是k，那就需要k\*k的矩陣，而參數越多越容易造成overfitting。

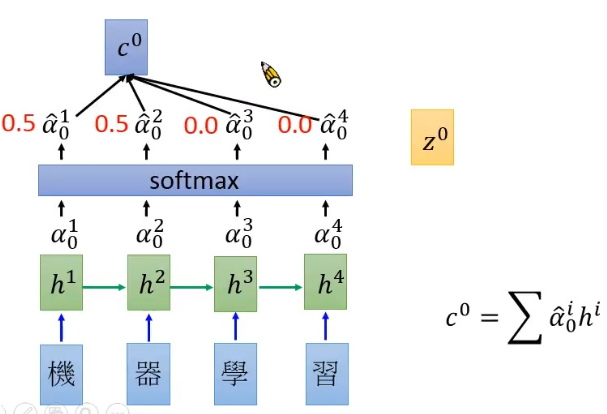
第二類attention-based model相較於RRN / LSTM的好處之一是當memory增加時，模型參數並不會增加。

接著以翻譯為例說明attention-based model的運作。



和seq-to-seq model相同，一開始都是一個encode RNN，但是這邊不期待最後輸出的hidden layer output vector能代表整個句子的資訊。

上圖中，為輸入字母的向量，接著初始化，為RNN的初始memory。接著會有一個match function，此match function可以有不同做法，例如cosine similarity(內積)。



上圖為cell state計算過程，其中為match function的輸出。將此輸出經過softmax後得到機率，並將機率當作是權重乘上h，即，得到cell state。接著將cell state送進RNN decoder得到即翻譯結果，如下圖所示

*一張含有 文字, 行, 字型 的圖片

自動產生的描述*

整體流程圖如下圖

一張含有 文字, 字型, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

1. Reference

[1] Fredrick Lee “Attention in Text : 注意力機制” <https://medium.com/@fredericklee_73485/attention-in-text-%E6%B3%A8%E6%84%8F%E5%8A%9B%E6%A9%9F%E5%88%B6-bc12e88f6c26>

[2] Rice Yang “RNN, LSTM, GRU之間的原理與差異” <https://u9534056.medium.com/rnn-lstm-gru%E4%B9%8B%E9%96%93%E7%9A%84%E5%8E%9F%E7%90%86%E8%88%87%E5%B7%AE%E7%95%B0-23eba88afa1e>

[3] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. <https://chat.openai.com/>

[4] MMChiou “Attention-based Model (Prof. 李宏毅)” <https://mmchiou.gitbooks.io/ai_gc_methodology_2018_v1-private/content/attention-based-model-li-hong-yi-jiao-638829/attention-based-model-prof-li-hong-6bc529.html>