深度學習

HW10

學號 : B103012002

姓名 : 林凡皓

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. 額外嘗試

* 重新訓練LSTM :

剛才訓練出來的LSTM表現不太好，因此我嘗試調整參數，重新訓練LSTM。

根據剛才的loss curve可以發現到，在訓練完成附近loss還有在下降，因此我猜測加大epochs數量可以有效提升模型表現。我將epochs調整成300後重新訓練，新的loss curve如下

一張含有 繪圖, 螢幕擷取畫面, 行, 圖表 的圖片

自動產生的描述

可以看到經過調整後，loss可以來到趨近於0的程度(實際值為0.03)。查看對training data以及validation data的表現

1. Training data :

一張含有 人員, 人的臉孔, 服裝, 食物 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, 室內 的圖片

自動產生的描述

由部份結果可以看出，對於training data來說，模型可以很準確的採樣。

1. Validation data :

一張含有 文字, 戶外, 告示牌, 天空 的圖片

自動產生的描述

一張含有 服裝, 樹狀, 人員, 戶外 的圖片

自動產生的描述

對於validation data來說，模型表現就比較差了，可能有overfitting的狀況發生。

* 時間序列AI :

由於這是我第一次接觸時間序列的模型，因此我希望可以多加熟悉相關概念。

首先先了解神經網路與地回神經網路的差別。

神經網路中會儲存關於特定問題的許多特徵(權重)。當我們要使用訓練好的神經網路來預測時，只要把一筆新的資料溜進去，神經網路會把資料分解成包含特徵的訊息。這些訊息會在神經網路內部傳遞並刺激神經網路，而神經網路會透過這些刺激來分析訊息所包含特徵，並得到答案。但是這個答案與時間無關，因為訓練好的神經網路每個神經元儲存的權重都已經固定住了，因此相同資料不論丟多少次都會得到相同答案。

RNN在這之上做出改進。每個神經元除了儲存特徵之外，還有一個儲存歷史資訊的hidden state。當我們進行預測時，，每個神經元的權重依然被固定住，但是hidden state卻沒有被固定住，讓神經網路可以根據輸入資料的變化調整輸出，得到一個相關的結果。

RNN的致命缺點在於很容易梯度消失，主要原因在於歷史資訊過長。因此發展出LSTM與GRU。這兩種模型都是在hidden state上動手腳來解決梯度消失的問題。

LSTM (Long short-term memory)多了input gate、forget gate、output gate。Input gate決定當前輸入是否要被記憶，forget gate決定是否遺忘先前的hidden state，output state決定當前所得之輸出要放多少進到hidden state。這三個gate的加入可以讓一些不重要的資訊被忽略，進而讓梯度可以順利進行遠距離傳播。

GRU (Gated recurrent unit)有reset gate與update gate。Reset gate決定是否要遺忘先前的hidden state，update gate決定先前的hidden state留下來的比例。

* Attention :

雖然說這次作業沒有實做到attention的部分，但是attention也是一種優化RNN的概念。

Attention-based model其實就是一個相似性的度量，當前的輸入與目標狀態越相近，那當前入的權重就會越大，代表說當前的輸出更加依賴當前的輸入。

attention架構大致如下

一張含有 文字, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

計算Query和key的相似度，常見的計算相似度的方法如下 :

1. 求兩者向量dot product 
2. 求兩者向量cosine相似性一張含有 文字, 字型, 印刷術, 白色 的圖片

   自動產生的描述
3. 引入額外神經網路求值

然後將所有對應的相似度與word value做相乘後相加，得到attention 

引入attention的RNN架構如下一張含有 文字, 螢幕擷取畫面, 圖表, 方案 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/>