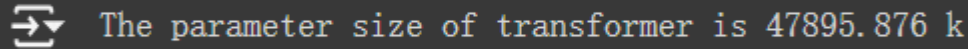


Deep learning hw3

109610025 陳品妍

Part 1:

Parameter size:

A terminal window with a dark background and light green text. It shows a message with a double arrow icon: "The parameter size of transformer is 47895.876 k".

```
➡ The parameter size of transformer is 47895.876 k
```

Accuracy:

```
Epoch: 9, Train loss: 3.470, Val loss: 5.104, Val Acc: 0.182, Epoch time = 265.124s
(model saved)
Epoch: 10, Train loss: 3.307, Val loss: 5.157, Val Acc: 0.191, Epoch time = 265.008s
(model saved)
Epoch: 11, Train loss: 3.166, Val loss: 5.239, Val Acc: 0.205, Epoch time = 265.202s
(model saved)
Epoch: 12, Train loss: 3.040, Val loss: 5.294, Val Acc: 0.185, Epoch time = 265.068s
(model saved)
Epoch: 13, Train loss: 2.927, Val loss: 5.331, Val Acc: 0.191, Epoch time = 265.020s
(model saved)
Epoch: 14, Train loss: 2.827, Val loss: 5.400, Val Acc: 0.184, Epoch time = 265.180s
(model saved)
Epoch: 15, Train loss: 2.733, Val loss: 5.457, Val Acc: 0.180, Epoch time = 265.400s
(model saved)
Epoch: 16, Train loss: 2.647, Val loss: 5.523, Val Acc: 0.176, Epoch time = 265.159s
(model saved)
Epoch: 17, Train loss: 2.574, Val loss: 5.546, Val Acc: 0.182, Epoch time = 265.199s
(model saved)
Epoch: 18, Train loss: 2.500, Val loss: 5.603, Val Acc: 0.181, Epoch time = 265.448s
(model saved)
```

Best validation accuracy: 0.205(epoch6)

Inference:

```
Input:      : 你好，欢迎来到中国
Prediction   : Hello, welcome you to China.
Ground truth : Hello, Welcome to China
Bleu Score (1gram): 0.6000000238418579
Bleu Score (2gram): 0.3872983455657959
Bleu Score (3gram): 0.0
Bleu Score (4gram): 0.0
```

```
Input:      : 早上好，很高兴见到你
Prediction  : Good morning, very happy to see you.
Ground truth: Good Morning, nice to meet you
Bleu Score (1gram): 0.4285714328289032
Bleu Score (2gram): 0.26726123690605164
Bleu Score (3gram): 0.0
Bleu Score (4gram): 0.0
```

```
Input:      : 祝您有个美好的一天
Prediction  : You have a good day.
Ground truth: Have a nice day
Bleu Score (1gram): 0.4000000059604645
Bleu Score (2gram): 0.3162277638912201
Bleu Score (3gram): 0.0
Bleu Score (4gram): 0.0
```

使用同個 model 只是後來有修改了 score 計算的問題(有同學在 fb 提到 score 的分數會因為算到 padding 的而被拉低)，不過這樣 inference 感覺變差:

```
Epoch: 10, Train loss: 3.307, Val loss: 5.139, Val Acc: 0.308, Epoch time = 243.469s
(model saved)
Epoch: 11, Train loss: 3.167, Val loss: 5.216, Val Acc: 0.307, Epoch time = 243.204s
(model saved)
Epoch: 12, Train loss: 3.041, Val loss: 5.300, Val Acc: 0.307, Epoch time = 243.300s
(model saved)
Epoch: 13, Train loss: 2.927, Val loss: 5.337, Val Acc: 0.309, Epoch time = 243.481s
(model saved)
Epoch: 14, Train loss: 2.825, Val loss: 5.396, Val Acc: 0.309, Epoch time = 243.501s
(model saved)
Epoch: 15, Train loss: 2.731, Val loss: 5.446, Val Acc: 0.308, Epoch time = 243.136s
(model saved)
Epoch: 16, Train loss: 2.649, Val loss: 5.504, Val Acc: 0.307, Epoch time = 243.421s
(model saved)
Epoch: 17, Train loss: 2.572, Val loss: 5.548, Val Acc: 0.309, Epoch time = 243.528s
(model saved)
Epoch: 18, Train loss: 2.500, Val loss: 5.611, Val Acc: 0.307, Epoch time = 243.619s
(model saved)
Epoch: 19, Train loss: 2.433, Val loss: 5.668, Val Acc: 0.304, Epoch time = 243.584s
(model saved)
Epoch: 20, Train loss: 2.375, Val loss: 5.707, Val Acc: 0.304, Epoch time = 243.294s
(model saved)
```

Best accuracy: 0.309

```
Input:           : 你好, 欢迎来到中国
Prediction        : Hello, welcome you to China.
Ground truth     : Hello, Welcome to China
Bleu Score (1gram): 0.6000000238418579
Bleu Score (2gram): 0.3872983455657959
Bleu Score (3gram): 0.0
Bleu Score (4gram): 0.0
```

```
Input:           : 早上好, 很高兴见到你
Prediction        : Good morning, I see you.
Ground truth     : Good Morning, nice to meet you
Bleu Score (1gram): 0.32749229669570923
Bleu Score (2gram): 0.2589053809642792
Bleu Score (3gram): 0.0
Bleu Score (4gram): 0.0
```

```
Input:           : 祝您有个美好的一天
Prediction        : I wish you have a good morning.
Ground truth     : Have a nice day
Bleu Score (1gram): 0.2857142984867096
Bleu Score (2gram): 0.2182179093360901
Bleu Score (3gram): 0.0
Bleu Score (4gram): 0.0
```

Inference 效果較第一版差了一點。

Jupyter 檔案繳交第二版檔案

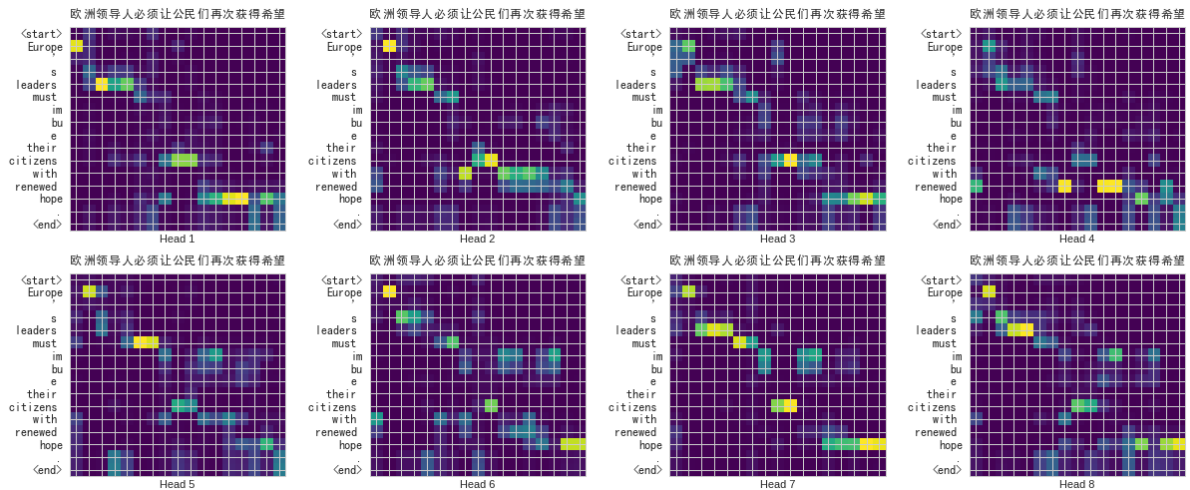
Part 2:

The structure of Transformer I implemented:

1. Dmodel = 512(emb_size=512):

可以給提供模型更大的彈性，使之可以同時關注在不同位子的 subword 的不同 subspace 下的 representation。可以提供足夠大的空間來捕捉詞語間的語義訊息，同時也部會使模型過於龐大以至於無法訓練。嵌入維度大能幫助模型可以理解模型中細緻的語意關係，特別針對句子結構較長的情況下。

2. NUM_HEAD=8:(論文中選擇 8)



圖片來源: <https://leemeng.tw/neural-machine-translation-with-transformer-and-tensorflow2.html>

可以發現不同 head 關注的東西不一樣，例如 head4 時在生成「們」與「再」特別關注 renewed，而 head8 則在「領導」與「希望」時分別關注 leaders 與 hope。

在網路上看到很多人針對 transformer 的 NUM_HEAD 都寫 8，原因在於可以幫助模型從不同的角度理解上下文，也可以學習語句中不同語意之間的關聯，尤其是多義詞。透過關注不同 subspace 的 subword 可以生成更好的結果。

3. FFN_HID_DIM=2048:

隱藏層的大維度可以增加模型的非線性轉換能力，針對語言這種數據複雜的模式，使用較大的隱藏層能夠幫助模型處理語言的特徵，提高結果的準確度。(此處參考論文中 d_model 為 512，dff 則為 4 倍的 2048)

4. NUM_ENCODER_LAYER = 1, NUM_DECODER_LAYER = 1:

沒有像論文中選擇 6 層，而選擇 1 的原因在於，訓練的時間限制以及初步試出來的模型取得不錯的效果，但因為 GPU 用量限制，常常在訓練到中間 EPOCH 時就中斷，因此沒有再加大 LAYER 數，平衡運算資源與時間的消耗。

5. DROPOUT = 0.2:

原本 DROPOUT 設為 0.1，但實驗過程中發現，TRAIN_LOSS 效果不錯然而 VALIDATION_LOSS 卻還是很高，因此有把 DROPOUT 加大防止 overfitting。

Encoder 和 decoder 裡的 layer normalization:

原本因為還沒發現計算 score 有錯誤時，想要增加收斂速度有試過 pre-layer normalization，在每個輸入之前進行 normalization 再進行 residual connection，發現這樣收斂速度比較快，而且可以用較大的 learning rate，但 validation 也是無法超過標準。後來找到問題之後，就沿用架構圖中的 post-layer normalization。在論文當中，post-layer normalization 可以穩定訓練過程，特別是較深的網路當中，也可以更好的處理梯度消失的問題。但缺點就是會導致訓練速度較慢，需要更多的 epoch。且根據論文的發現，研究人員發現 post-layer

normalization 可以有較好的 performance 在處理生成文章的任務當中。另外，也可以減緩不穩定的訓練特性。

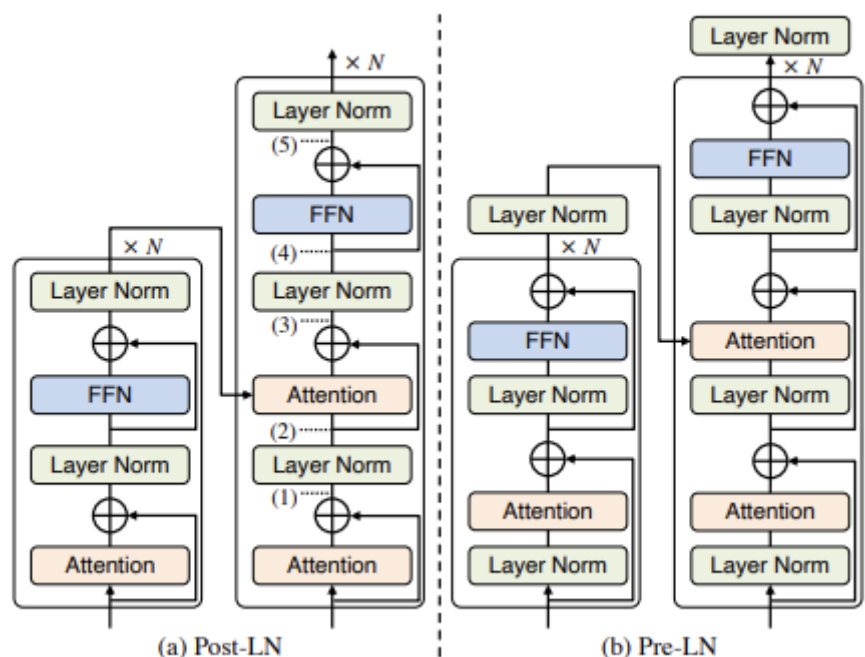


Table 1: BLEU scores of each method on WMT newstest2010-2016 and their averages.

Method	2010	2011	2012	2013	2014	2015	2016	Average
Enc-Dec: 6L-6L								
Post-LN	24.27	22.06	22.43	26.11	27.13	29.70	34.40	26.59
Pre-LN	24.03	21.77	22.08	25.63	26.27	29.07	33.84	26.10
B2T connection	24.12	21.93	22.29	26.31	26.84	29.48	34.73	26.53

Post-layer normalization 與 pre-layer normalization 示意圖以及論文中的實驗結果 (encoder/decoder 各 6 層)

圖片來源:

<https://arxiv.org/pdf/2206.00330v1#:~:text=There%20are%20currently%20two%20major%20layer%20normalization%20positions,each%20residual%20connection.%20The%20original%20Transformer%20employs%20Post-LN.>

learning rate scheduler and other strategies:

實驗中並沒有使用這個策略，原本想說等 inference 出來結果如何再決定是否增加 warmup 以及 learning rate scheduler，但神奇地發現第一版出來結果還蠻不錯的，所以為了減少 super parameter 的複雜度後來就沒有再另外寫。此次 optimizer 跟以往一樣選擇 torch.optim.Adam，設置 learning rate= 5e-4，發現這個設定的效果可以介於 1e-3 以及 1e-4 間。前者的 validation accuracy 一直無法

下降，研判是因為 lr 太大導致找不到最低點來收斂。而 $1e-4$ 則是收斂速度有些慢，需要加大 `epoch` 數但又會因為 `gpu` 的關係暫停。`Betas=(0.9,0.98)`設置了兩個指數衰減率，用於估計一階和二階矩，幫助梯度更新。最後 `eps = 1e-9` 防止計算中的分母為零，提升數值穩定性。

Difficulty:

我覺得最難的部分是 `multiheadattention` 以及 `create_mask` 部分，加上剛開始因為有點搞不清楚 `mask` 的 `size`，所以一直都有 `mismatch` 的問題。後期，改正之後，發現即使 `model` 可以訓練，但表現很糟糕。所以來回看網路上的資料修正自己的 `mask` 生成方式以及維度等問題(之後發現 `mask` 即使寫對會大大影響模型的表現)。改正之後，又發現 `src` 的 `size` 不是預期的，後來才發現是自己 `padding` 的問題。但之後又一直被 `validation accuracy` 太低的問題給困擾，因為想說等 `validation accuracy` 修好再去用 `inference`，但沒想到就是這個想法絆住自己。之後去網路上看網友寫的架構也都跟我相似，反正就是不知道到底哪裡有問題。後來我在清交二手拍請人幫我看架構，結果每個人都說沒問題，而且他們嘗試改動參數也都沒啥變化，真的懷疑到天邊。後來隨意的在 `instagram` 發了類似生無可戀的限動，結果有一個朋友也跟我有一樣的困擾，於是她詢問我是否有做 `inference`。後來我想說那做做看，反正也不知道該怎麼辦，最後竟然發現效果比預期的好，直接嚇到。然後就在 `fb` 問助教到底是怎麼回事，原本懷疑 `mask` 沒有針對 `batch` 裡的不同 `sequence` 寫，但後來用 `for` 迴圈明確寫出來，`validation accuracy` 也沒什麼改善。後來，當初有跟我一樣情況的同學說可能是 `score` 那邊有問題，我就發現因為當初把助教 `code train` 的地方 `transpose` 改掉導致在計算 `score` 時拿錯了 `batch_size` 拿成了 `sequence length`。改了之後，就正常了。

另外一個難處是因為 `gpu` 的資源限制所以常常在訓練到一半的時候就停止了，當下真的很想罵髒話。最後還是跟朋友借帳號，總共用了 4 個帳號來完成此次的 `lab`，這也是原因為何沒有加大 `decoder` 和 `encoder` 的層數。因為 `hidden layer` 跟 `emb_size` 已經加大了訓練時間，所以就想說不用再增加，況且結果還算不錯。

Reference:

<https://leemeng.tw/neural-machine-translation-with-transformer-and-tensorflow2.html>

<https://stackoverflow.com/questions/65343377/adam-optimizer-with-warmup-on-pytorch> (之後要用 `warmup` 的話)

<https://arxiv.org/abs/1607.06450>