GAI Project 2a 資訊 113 F74096027 吳瀚惟

1. Model Analysis

我設計的模型是基於遞迴神經網路(RNN)，使用了**長短期記憶網路(LSTM)**。LSTM 是一種特殊的 RNN，能夠在長 sequence 中有效學習相關概念，是專為**處理序列數據**而設計的，特別適合處理本次 project 中的四則運算式的結果生成任務。我上網搜尋資料，經過吸收與理解，明白 LSTM 的核心主要是利用它的 cell state 與三個 gates 的機制來密切合作以有效保存並控制訊息的傳遞，這三個 gates 如以下所示：

(1) Forget Gate: 作用是決定要從 cell state 中遺忘什麼訊息，功能是通過一個 sigmoid 層來決定哪些是要被遺忘的訊息，輸出 0 表完全丟棄，輸出 1 則是完全保留。

(2) Input Gate: 作用是決定哪些訊息要被儲存在 cell state 中。功能是一個 sigmoid 層決定要更新哪些值，以及一個 tanh 層創造一個新的候選值向量，而這個向量將被加到狀態中。

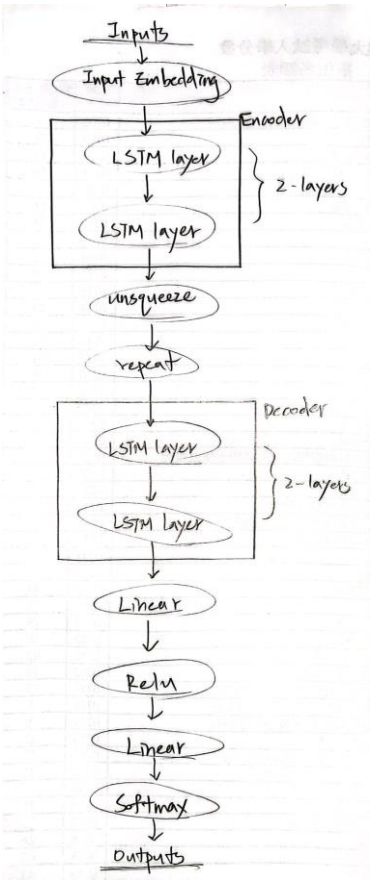
(3) Output Gate: 作用是根據當前的 cell state 和輸入去決定要輸出什麼，功能大致為：一個 sigmoid 層來判斷哪些 cell state 用於輸出，然後將 cell state 通過 tanh 再乘上 sigmoid 的輸出，最後決定輸出什麼。

而我的 model design 如下圖所示。首先是針對 input 做 **embedding**，

在 ArithmeticDataset 這個 class 中，除了 split data 我還實作了 tokenize 的功能，將每個可能出現的符號建立成一個 char_id_list，讓四則運算式轉換為自然語言的形式，以便於做後續的訓練。

接下來是 **Encoder**，我使用雙層 LSTM 循環層來處理輸入序列，一步步更新其神經網路層的內部狀態，最後以 last time step 的 hidden 狀態作為整個序列的編碼結果(在 GRU 的情況下是 last time step 的輸出)，做 unsqueeze 和 repeat 處理後被送入解碼器。

Decoder 收到處理過的上下文向量，在雙層單向 LSTM 循環層的結構下按順序進行解碼，最後經過兩層全連接層(線性變換)將輸出轉換為最終的預測序列。



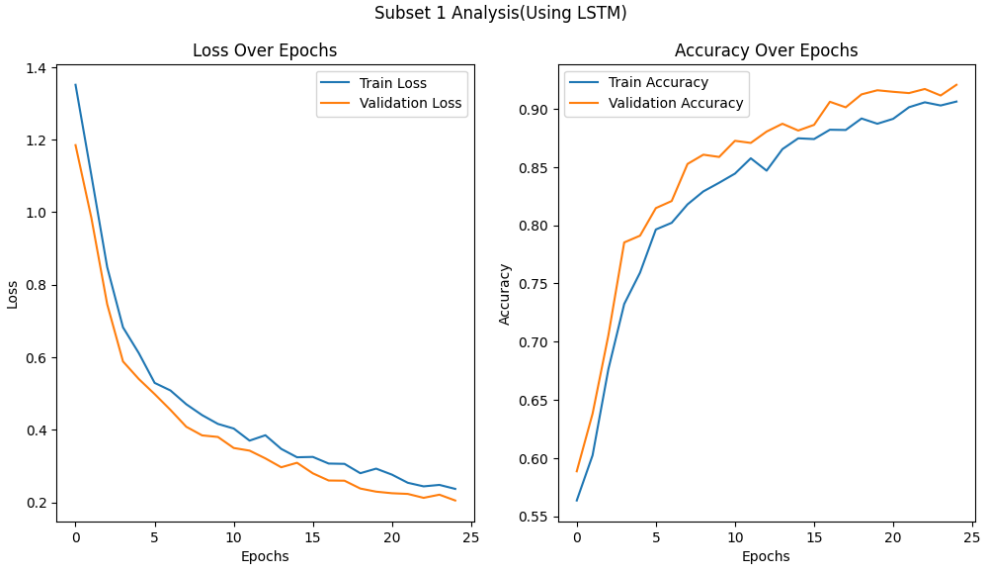
關於 loss function，我選擇的是 **CrossEntropyLoss**，因為我將本次 project 視為**分類問題**，這需要從一定的輸出空間中選擇出正確的類別。在實作的時候，我遇到了**維度錯誤**的問題，在研究和調整過後才明白輸出 logits 或 label 的形狀一定要正確，也要確保輸出沒有被錯誤壓縮或擴張。

至於驗證的部分，我將在第二點討論使用不同資料集訓練出來的結果，做進一步的分析。

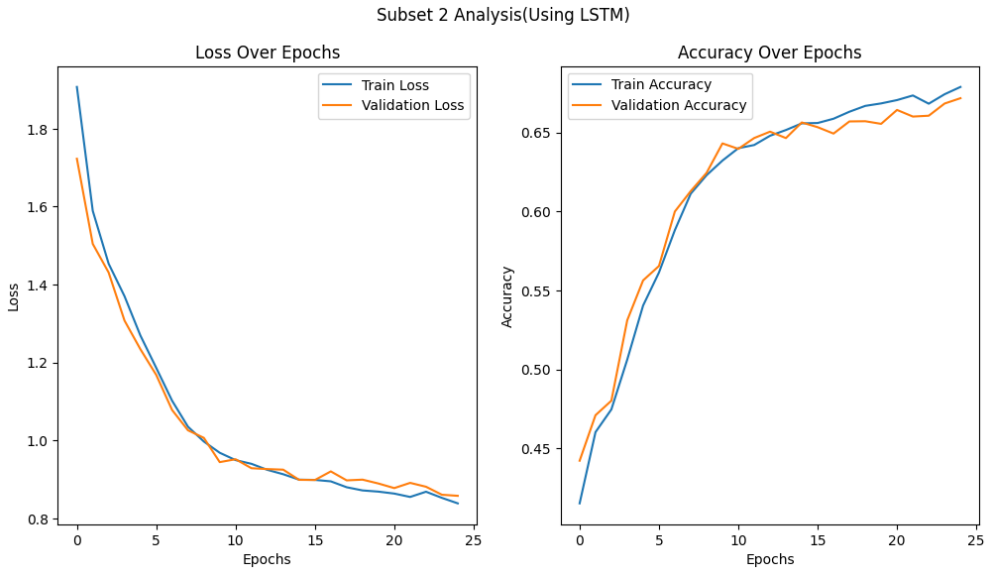
2. Dataset Analysis

我生成的 datasets 一共有 **3 種**，分別是以不同的規則從原始資料集進行篩選。以下為針對三種資料集的基本介紹：

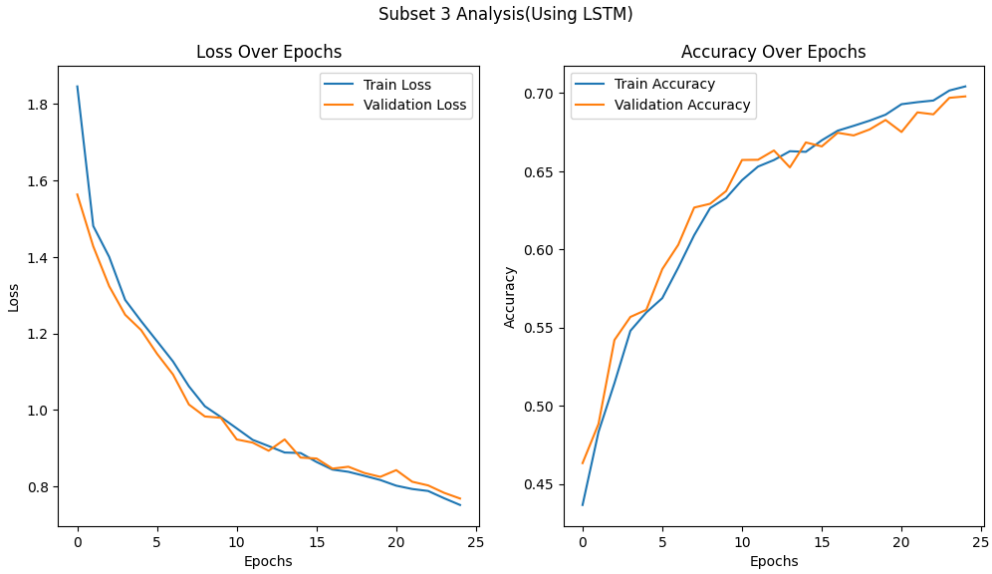
(1) Subset 1: 取僅含單個 digit (0-9)的運算式。經過自定義的 function，我篩選出 21300 組符合條件的資料集。其中切割九成作為訓練資料集，一成作為驗證資料集。訓練與驗證的 loss 和 accuracy 如下圖所示，可以看到**最終的 loss 落在 0.2 附近，而準確率已經超過九成**。以我的理解，在資料集不算太大的情況下，accuracy 之所以能達到不錯的水準是因為這是僅有單個 digit 的四則運算，算式中的數值以及可能的輸出值也相對於完整資料集來的小，因而較易於訓練。加上在這次訓練中，我是以**所有符合”僅含單個 digit 運算式”特徵的資料集**來進行訓練與驗證，並沒有先做資料縮小，因此得到的結果較佳。



(2) Subset 2: 取僅含兩個 digit(10-49)的運算式。經過自定義的 function，我篩選出 1348800 組符合條件的資料集。由於資料集很大，加上我擁有的運算資源有限，我將這些資料先行縮小，隨機取其中的 21075 組，接著再切割九成作為訓練資料集，一成作為驗證資料集。訓練與驗證的 loss 和 accuracy 如下圖所示，可以看到**最終的 loss 落在 0.85 附近，而準確率則是落在 0.67 左右。**我對這個結果的理解是因為兩位數運算的算式本身就比較複雜，輸出的數據序列相對於單 digit 運算式的輸出範圍也較大，因此比較不利於模型訓練與推測出正確的結果。若使用完整的 subset 2 進行訓練或許會得到不錯的效果，在我擁有更好的運算資源後，我會利用時間再來測試看看上述的假設是否正確。

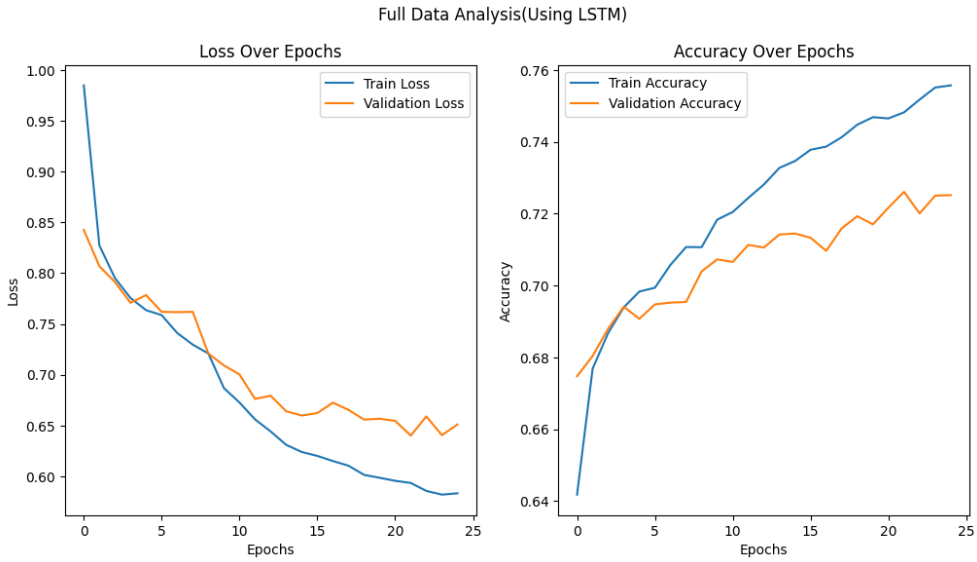


(3) Subset 3: 排除所有包含’1’的運算式。經過自定義的 function，我篩選出 983664 組符合條件的資料集。由於資料集很大，加上我擁有的運算資源有限，我將這些資料先行縮小，隨機取其中的 20288 組，接著再切割九成作為訓練資料集，一成作為驗證資料集。訓練與驗證的 loss 和 accuracy 如下圖所示，可以看到**最終的 loss 落在 0.78 附近，而準確率則是七成左右**。我的理解是，相對於 subset 2 的原資料集大小(1348800 筆)，subset 3 的資料集數目稍微少一些，同樣隨機取 20000 筆左右的情況下，模型能學到的特徵自然會比較多(20000/1348800 < 20000/983664)，因此 performance 也相較於上一點中的結果來的好一些(相對 loss 較低，accuracy 稍高)。



最終，為了更好的比對與理解訓練成果，我也有使用**原始資料集**做訓練。原始資料集一共有 2632500 筆，由於資料集龐大，加上我擁有的運算資源有限，我仍然將這些資料先行縮至 20000 多組，接著再切割九成作為訓練資料集，一成作為驗證資料集。訓練與驗證的 loss 和 accuracy 如下圖所示，可以看到**最終的 loss 落在 0.6-0.65 附近，而準確率則落在 0.71-0.76 左右**。從繪製的圖表中，我觀察到了兩個現象：**validation loss 和 validation accuracy 隨著 epochs 增加的變化率相對較低**，以及**它們與 train loss 和 train accuracy 存在一定的差距**。針對這兩點，我的推測是由於原始資料集很大，從其中的兩百多萬組隨機取兩萬多組再做切割，可能較容易遇到**訓練集與驗證集分布差異**的問題，導致驗證指標改善不明顯，以及兩個 accuracy 之間存在差距的現象。因此，我相信若使用”完整的完整資料集”進行訓練理應會得到更好的效果，在我

擁有更好的運算資源後，我也會再利用時間，測試看看上述的假設是否正確。



3. Discussion

不同的 learning rates 會影響模型的訓練速度和穩定性。我上網搜尋資料並實測，了解到較低的 lr，如 0.0001，雖然有助於達到更穩定的收斂，但是訓練的速度真的很慢；較高的 lr，如 0.01，雖然相對快速但可能會導致訓練過程中出現波動或是不收斂。多次的實測下，最後我選擇了 0.001 的學習率，套用於 **AdamW** 權重更新演算法。

此外，我也針對 batch size 做了實測調整。我發現較大的 batch size 有助於提供更穩定的梯度估計，但會增加記憶體的負擔並且讓訓練速度變慢；較小的 batch size 則可以幫助模型更快的更新權重，但可能使訓練過程的損失波動變大。

經過調整，我最終使用的超參數配置為：

 # batch_size = 256

 # epochs = 25

 # embed_dim = 256

 # hidden_dim = 256

 # lr = 0.001

 # grad_clip = 2 ### 用於防止梯度爆炸

我在這次 project 中所使用的 model 是 LSTM，bonus 中所用的是 GRU。兩者因為都能**有效地捕捉長期依賴關係**，而被廣泛的應用於**處理序列數據**。他們可以**學習到數字間操作的規則**，所以適合進行本次 project 中的算術運算任務。以模型架構的細節做討論，兩者都是 **encoder-decoder(seq-2-seq)** 的架構，模組化的設計讓模型易於訓練。在 encoder 中我使用雙層單向的 LSTM，幫助模型有效學習四則運算式的順序特徵，並將其映射成向量模式；在 decoder 中，我也是使用雙層 LSTM，處理 encoder 輸出後單向的為不同位置進行解碼；最後經過兩個線性層做降維以形成各位置的機率分布。

[註] 關於這次的訓練任務，我認為有幾個可以改進的部分，第一是嘗試增加 epoch 數，因為從第二點中的四張圖片來看，輸出的結果似乎還沒有達到

完全收斂，在時間允許的情況下我會再嘗試更多 epoch 數的訓練結果；第二點如前文中提到，我可以嘗試尋找更好的運算資源，利用完整的 subsets 進行訓練與驗證，觀察結果，去思考並驗證我在 report 中所做的推測是否合理。

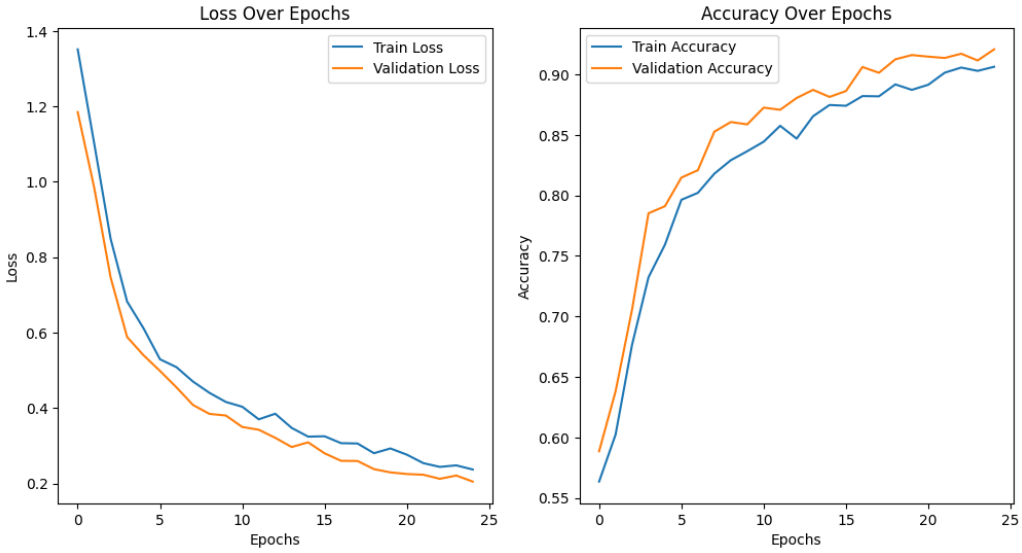
4. Bonus

在本項分析中，我是以 subset 1 --- **取僅含單個 digit (0-9)的運算式**作為比較 LSTM 以及 GRU 兩個 model 的實驗資料集，我也給予他們完全相同的超參數配置，嘗試直觀的對比模型之間的 performance。透過下頁兩張訓練成果圖，我將簡短說明我觀察到的現象：

(1) **GRU 訓練一個 epoch 的時間較 LSTM 來的少。**我上網搜尋後，了解到這樣的情況是由於 GRU 的結構較 LSTM 簡單(GRU 只有兩個 gate，沒有單獨的 cell state；而 LSTM 有三個 gate 加上單獨的 cell state)，這才導致 GRU 的訓練速度較快，對計算資源的要求也沒那麼高。

(2) 兩者在 loss 和 accuracy 的表現上差不多。理論上 LSTM 在處理長序列和複雜任務時應該會表現得稍微好一點，performance 差不多的原因可能是因為 subset 1 資料集較小，或是由於這次的任務對於兩個模型而言不算太複雜(?)，因此在結構差異的情況下仍達到差距不大的 loss 和 accuracy 表現。

Subset 1 Analysis(Using LSTM)



Bonus: Subset 1 Analysis(Using GRU)