Embedded system of Transformer-based TTS

Sian-Yi Chen

Advisors : Tay-Jyi Lin and Chingwei Yeh

Outline

Action item

- 完成 Transformer-based TTS 的嵌入式系統 (尚未完成)

Status report

- 先前進度
  - 透過中斷點紀錄所有經過的程式碼、傳遞的參數，但遇到以下兩個問題
    1. 程式編輯器無法進入transformer底層的程式，因此對於底層運作過程仍需理解
    2. 載入的參數檔(bin_file.ark)，使用的是kaldi特有的資料格式，因此對於要如何實作沒有想法
- 本周進度 (進行中)
  - 先前進度的解決方法
    1. 中斷點無法進入底層程式僅是因為編輯器預設的設定導致，將設定改為執行所有程式碼，而非自己的程式碼即可
    2. 參數檔為具有特定格式的binary檔，若要使用C實現，我認為過於耗費資源，後續判斷可以透過python將值取出，並使用陣列直接存取
  - TTS實作進度
    - 在學長的協助下已熟悉大部分的encoder架構，目前剩下embedding層ScaledPositionalEncoding中有alpha值與pe值仍須釐清
      - Embedding層：透過一張維度為[337, 384]的表，對input做編碼，已取得這張訓練完成的參數表
      - MultiHeadedAttention以及PositionwiseFeedFpeward中僅使用Linear、ReLU，Linear為轉置矩陣相乘
      - 最後使用LayerNorm來Normalization (計算與驗算結果放置附錄)
    - (已解決) 因pytorch高度模組化結構無法直接查看神經網路隱藏層的輸入輸出
    - 在pytorch中embedding有兩種函數，分別是「torch.nn.functional.embedding」與「torch.nn.Embedding」，在驗算時使用錯誤函數導致結果對不起來，目前在重新驗算中

附錄

LAYERNORM [LayerNorm — PyTorch 1.12 documentation]

CLASS `torch.nn.LayerNorm(normalized_shape, eps=1e-05, elementwise_affine=True, device=None, dtype=None)` [SOURCE]

Applies Layer Normalization over a mini-batch of inputs as described in the paper Layer Normalization

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

transform parameters of `normalized_shape` if `elementwise_affine` is `True`. The standard-deviation is calculated via the biased estimator, equivalent to *torch.var(input, unbiased=False)*.

其中variance預設使用帶有bias的estimator，使用的公式如下

[torch.var — PyTorch 1.12 documentation]

You probably know that the expectation of the unbiased estimator is

$$E\left[\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2\right] = \sigma^2$$

To obtain the expectation of the biased estimator we just have to multiply both sides by $(n-1)$ and divide them by n

$$E\left[\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2\right] = \sigma^2 \cdot \frac{n-1}{n}$$

[statistics - How to calculate the bias of the estimator for variance? - Mathematics Stack Exchange]

驗算：

若一矩陣為 [1, 1, 2, 4]
平均數為 (1+1+2+4) / 4 = 2
變異數為 (2-1)**2 + (2-1)**2 + (2-2)**2 + (2-4)**2 = 1.5

```
print(torch.var(torch.tensor([1.0, 1.0, 2.0, 4.0]), unbiased=False))
>>> tensor(1.5000)
```