Self-attention 模型來源-HUNG-YI LEE (李宏毅) MACHINE LEARNING
2023 SPRING: https://speech.ee.ntu.edu.tw/~hylee/ml/2023-spring.php

HW 4 3/17 Self-attention Kaggle, COOL, Gradescope: 04/07/2023 23:59
- 自注意力機制(Self-attention) (上)
- 自注意力機制(Self-attention) (下)

主題: Self-attention 語音識別

主題描述:

目標為多分類語音識別任務，從給定的語音資料中預測說話者的類別，構建和調整 Transformer 模型來達成目標。

引用的資料與程式碼敘述:

引用的資料:

Dataset 使用 VoxCeleb2

- 訓練集：56666 個已處理的帶標籤的音頻特徵。
- 測試集：4000 個已處理的無標籤音頻特徵（包括公共和私有部分）。
- 標籤：總共有 600 個類別，每個類別代表一個說話者。
- Link:https://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox2.html

程式碼敘述:

1. 從指定的鏈接下載文件，將文件合併成一個壓縮文件，解壓。

2. 定義了一個名為 `set_seed` 的函數，用於設置隨機種子，以確保代碼的可重複性； 定義`myDataset` 的自定義數據集類，用於加載音頻數據集。

3. 定義了一個函數 `get_dataloader`，用於生成訓練集和驗證集的數據加載器，返回加載器以及說話者的數量。

4. 定義了一個為 `Classifier` 的神經網絡模型，用於說話者識別任務；定義了函數 `get_cosine_schedule_with_warmup`，用於創建一個學習率調度器。

5. 定義了一個函數 `model_fn`，用於執行模型的前向傳播，並計算損失和準確率；定義了一個函數 `valid`，用於在驗證集上進行模型的驗證。

6. 引入必要模組後設定訓練所需的參數；main 函數負責數據加載、模型初始化、訓練和驗證過程。

7. 透過 parse_args 解析參數並執行主訓練函數。使用 tqdm 顯示訓練進度，在每次驗證後更新最佳模型參數。

自行修改的部分比對:

1.將 d_model=80 改為 d_model=512
2.將 nhead=2 改為 nhead=16
3.新增 self.encoder = nn.TransformerEncoder(self.encoder_layer, num_layers=6)
4.將 nn.Sigmoid()改動為使用 nn.ReLU()

```
[ ]  import torch
     import torch.nn as nn
     import torch.nn.functional as F


     class Classifier(nn.Module):                                    1
         def __init__(self, d_model=512, n_spks=600, dropout=0.1):  #80
             super().__init__()
             # Project the dimension of features from that of input into d_model.
             self.prenet = nn.Linear(40, d_model)
             # TODO:
             #     Change Transformer to Conformer.
             #     https://arxiv.org/abs/2005.08100
             self.encoder_layer = nn.TransformerEncoderLayer(                    2
                 d_model=d_model, dim_feedforward=256, nhead=16   #4
             )                                                                   新增
     3       self.encoder = nn.TransformerEncoder(self.encoder_layer, num_layers=6)    #4

             # Project the the dimension of features from d_model into speaker nums.
             self.pred_layer = nn.Sequential(
                 nn.Linear(d_model, d_model),
                 # nn.Sigmoid(),
     4           nn.ReLU(),                                                      新增改動
                 nn.Linear(d_model, n_spks),
             )
```

1.將 d_model=80 改為 d_model=512
解釋：d_model 是 Transformer 模型中的特徵維度。它表示在輸入特徵被投影到 Transformer 模型之前，將被轉換為 d_model 維度的空間。將 d_model 設置為 512，代表每個輸入特徵向量將被轉換為 512 維的表示。

2.d_model=d_model, dim_feedforward=256, nhead=16
解釋：nhead 是多頭注意力機制中的頭數。多頭注意力機制將注意力機制分成多個頭，允許模型在不同的子空間中學習不同的表示。將 nhead 設置為 16，代表注意力機制將被分成 16 個平行的頭。

3.self.encoder = nn.TransformerEncoder(self.encoder_layer, num_layers=6)
解釋：創建了一個 Transformer 編碼器，由 num_layers 層 Transformer 編碼器層組成。將 num_layers 設置為 6，表示 Transformer 編碼器由 6 層堆疊組成，每一層都是一個獨立的 Transformer 編碼器層。

4.將 nn.Sigmoid()改動為使用 nn.ReLU()

解釋：使用 nn.ReLU() 是為了避免梯度消失，提高計算效率。

nn.ReLU() 將所有正輸入保持不變，負輸入置零，因此梯度不會消失。使得 ReLU 能夠在深層網絡中有效地傳遞梯度，從而加速收斂。

nn.Sigmoid() 將輸入壓縮到 (0, 1) 的範圍內，這樣會導致在反向傳播過程中，梯度值可能變得非常小，進而導致梯度消失問題，特別是在深層神經網絡中。

nn.ReLU() 的計算非常簡單，只需判斷輸入是否大於零。
nn.Sigmoid() 需要計算指數函數，計算量更大。

原本模型(acc = 0.4834)

```
Train: 100% 2000/2000 [00:45<00:00, 43.60 step/s, accuracy=0.56, loss=2.07, step=52000]
Valid: 100% 5664/5667 [00:03<00:00, 1704.31 uttr/s, accuracy=0.47, loss=2.44]
Train: 100% 2000/2000 [00:45<00:00, 44.30 step/s, accuracy=0.47, loss=2.27, step=54000]
Valid: 100% 5664/5667 [00:03<00:00, 1416.50 uttr/s, accuracy=0.47, loss=2.45]
Train: 100% 2000/2000 [00:45<00:00, 44.31 step/s, accuracy=0.38, loss=2.47, step=56000]
Valid: 100% 5664/5667 [00:03<00:00, 1636.55 uttr/s, accuracy=0.47, loss=2.41]
Train: 100% 2000/2000 [00:46<00:00, 43.13 step/s, accuracy=0.53, loss=2.40, step=58000]
Valid: 100% 5664/5667 [00:03<00:00, 1733.87 uttr/s, accuracy=0.47, loss=2.42]
Train: 100% 2000/2000 [00:46<00:00, 43.38 step/s, accuracy=0.56, loss=1.91, step=6e+4]
Valid: 100% 5664/5667 [00:03<00:00, 1722.16 uttr/s, accuracy=0.48, loss=2.39]
Train:   0% 9/2000 [00:00<00:44, 44.52 step/s, accuracy=0.56, loss=2.01, step=6e+4]Step 60000, best model saved. (accuracy=0.4802)
Train: 100% 2000/2000 [00:46<00:00, 43.09 step/s, accuracy=0.38, loss=2.42, step=62000]
Valid: 100% 5664/5667 [00:03<00:00, 1652.15 uttr/s, accuracy=0.48, loss=2.36]
Train: 100% 2000/2000 [00:45<00:00, 43.60 step/s, accuracy=0.53, loss=2.07, step=64000]
Valid: 100% 5664/5667 [00:03<00:00, 1482.73 uttr/s, accuracy=0.48, loss=2.39]
Train: 100% 2000/2000 [00:47<00:00, 42.54 step/s, accuracy=0.53, loss=2.47, step=66000]
Valid: 100% 5664/5667 [00:03<00:00, 1674.74 uttr/s, accuracy=0.48, loss=2.40]
Train: 100% 2000/2000 [00:46<00:00, 43.00 step/s, accuracy=0.41, loss=2.61, step=68000]
Valid: 100% 5664/5667 [00:05<00:00, 990.67 uttr/s, accuracy=0.48, loss=2.37]
Train: 100% 2000/2000 [00:48<00:00, 41.57 step/s, accuracy=0.69, loss=1.81, step=7e+4]
Valid: 100% 5664/5667 [00:03<00:00, 1500.45 uttr/s, accuracy=0.47, loss=2.42]
Train:   0% 0/2000 [00:00<?, ? step/s]
Step 70000, best model saved. (accuracy=0.4834)
```

改動後模型(acc = 0.7599)

```
Train: 100% 2000/2000 [00:44<00:00, 44.52 step/s, accuracy=0.78, loss=0.65, step=52000]
Valid: 100% 5664/5667 [00:03<00:00, 1829.27 uttr/s, accuracy=0.74, loss=1.42]
Train: 100% 2000/2000 [00:46<00:00, 42.58 step/s, accuracy=0.91, loss=0.69, step=54000]
Valid: 100% 5664/5667 [00:03<00:00, 1436.91 uttr/s, accuracy=0.74, loss=1.40]
Train: 100% 2000/2000 [00:45<00:00, 44.17 step/s, accuracy=0.94, loss=0.54, step=56000]
Valid: 100% 5664/5667 [00:03<00:00, 1814.78 uttr/s, accuracy=0.74, loss=1.41]
Train: 100% 2000/2000 [00:47<00:00, 42.10 step/s, accuracy=0.97, loss=0.17, step=58000]
Valid: 100% 5664/5667 [00:03<00:00, 1465.79 uttr/s, accuracy=0.75, loss=1.42]
Train: 100% 2000/2000 [00:44<00:00, 44.96 step/s, accuracy=0.84, loss=0.61, step=6e+4]
Valid: 100% 5664/5667 [00:03<00:00, 1739.01 uttr/s, accuracy=0.75, loss=1.41]
Train:   0% 6/2000 [00:00<01:43, 19.28 step/s, accuracy=0.81, loss=0.96, step=6e+4]Step 60000, best model saved. (accuracy=0.7528)
Train: 100% 2000/2000 [00:47<00:00, 42.42 step/s, accuracy=0.91, loss=0.49, step=62000]
Valid: 100% 5664/5667 [00:03<00:00, 1442.16 uttr/s, accuracy=0.75, loss=1.41]
Train: 100% 2000/2000 [00:45<00:00, 44.04 step/s, accuracy=0.94, loss=0.43, step=64000]
Valid: 100% 5664/5667 [00:03<00:00, 1822.84 uttr/s, accuracy=0.76, loss=1.39]
Train: 100% 2000/2000 [00:47<00:00, 42.13 step/s, accuracy=0.91, loss=0.21, step=66000]
Valid: 100% 5664/5667 [00:03<00:00, 1512.12 uttr/s, accuracy=0.76, loss=1.36]
Train: 100% 2000/2000 [00:44<00:00, 44.94 step/s, accuracy=0.91, loss=0.29, step=68000]
Valid: 100% 5664/5667 [00:03<00:00, 1632.45 uttr/s, accuracy=0.76, loss=1.38]
Train: 100% 2000/2000 [00:47<00:00, 42.52 step/s, accuracy=0.91, loss=0.31, step=7e+4]
Valid: 100% 5664/5667 [00:03<00:00, 1499.54 uttr/s, accuracy=0.76, loss=1.38]
Train:   0% 0/2000 [00:00<?, ? step/s]
Step 70000, best model saved. (accuracy=0.7599)
```

心得敘述:

通過調整 Transformer 模型參數，包括將 d_model 從 80 改為 512，nhead 從 2 改為 16，新增 6 層 Transformer 編碼器，並將激活函數從 Sigmoid 改為 ReLU，提升了模型性能，準確率從 0.4834 顯著提升到 0.7599。看到模型準確率提高蠻多的，讓我感到非常開心，未來希望能找到不同方式繼續優化模型，進一步提升準確率。