

EAI lab 3

Human Pose Estimation

數據所 RE6121011 徐仁瓏

1. Code Description

函式 merge 接收三個參數，一個為骨架（subset） ，一個為最小的關節點個數（min_num_body_parts） ，一個為最小平均分數（min_score） ，透過以下合併和刪除的程式碼求出最終需要的人體骨架（subset） 。

程式碼分為兩個部分，一個為合併，另一個為刪除。

- 合併：

- 首先建立一個列表 delete_idx 記錄要刪除的索引，接著開始尋找兩兩個骨架子集（subset）中有沒有重疊的部位（part） ，從第 i 個子集往後與第 j 個子集比較，即第 1 個子集與第 2 個之後的子集比較，第 2 個子集再與第 3 個之後的子集比較，如果有則把他們存放到列表 common_part 中。
- 若 common_part 中有值，表示兩兩骨架中有重疊的部分，則將後面的子集（即第 j 個）有偵測到關節的部分合併到前面的子集（即第 i 個）中，並將分數和關節點個數合併上去，最後將這個（第 j 個）子集記錄為要刪除的子集，即存放到 delete_idx 中。

```
#---merge---
# Merge the limbs in the subset
# score : score
# parts_num : How many limbs are in the subset
#####
delete_idx = []

# 遍歷 subset 中的每個骨架子集，將相互重疊的部分合併
for i in range(len(subset)):
    for j in range(i + 1, len(subset)):
        # 檢查是否有相同的部位（即如果某一部位在兩個子集中都存在）
        common_parts = [part for part in range(len(subset[i]) - 2)
                        if subset[i][part] >= 0 and subset[i][part] == subset[j][part]]

        if len(common_parts) > 0: # 如果有重疊的部分
            for k in range(len(subset[i]) - 2): # 遍歷所有關節
                if subset[i][k] < 0 and subset[j][k] >= 0: # 若 i 缺少某關節但 j 有該關節
                    subset[i][k] = subset[j][k]
            # 合併總分數和部件數量
            subset[i][-2] += subset[j][-2] # 合併分數
            subset[i][-1] += subset[j][-1] # 合併部件數
            delete_idx.append(j) # 記錄需要刪除的索引
```

- 刪除：

- 上述已累計在 `delete_idx` 中累計好要刪除的骨架的索引，接著再檢查有沒有不符合資格的骨架，即關節點數量少於 5 個或平均分數小於 0.4，若有此情形發生也將此骨架放入 `delete_idx` 中等待刪除
- 最後透過已經記錄好要刪除的骨架索引 `delete_idx` 來刪除不需要的骨架，剩下的即為我們需要的骨架（subset）。

```
# after merge
#---delete---
# Delete the non-compliant subset
# 1. parts_num < 5
# 2. Average score(score / parts_num) < 0.4
#####
for i in range(len(subset)):
    if subset[i][-1] < min_num_body_parts or (subset[i][-2] / subset[i][-1]) < min_score:
        delete_idx.append(i)

subset = np.delete(subset, delete_idx, axis=0)
```

2. Output Subsets

- Image1

```
subset:
[[ 0.          1.          2.          3.          4.          5.
   6.          7.          8.          9.          10.         11.
  12.         13.         14.         15.         17.         18.
  41.13863325 38.          ]]
```

- Image2

```
subset:
[[ 3.          8.          13.         18.         22.         28.
   33.         37.         44.         47.         52.         59.
  62.         67.         73.         78.         83.         87.
  44.87913832 40.          ]
 [ 0.          5.          10.         15.         21.         25.
   30.         35.         40.         45.         50.         55.
  60.         -1.         70.         -1.         80.         85.
  38.39658982 32.          ]
 [ 2.          7.          12.         16.         20.         27.
   32.         36.         42.         49.         54.         57.
  64.         69.         72.         77.         82.         -1.
  37.66937459 34.          ]
 [ 1.          6.          11.         17.         -1.         26.
   31.         -1.         41.         46.         51.         56.
  61.         65.         71.         76.         81.         86.
  42.08062305 34.          ]
 [ 4.          9.          14.         19.         24.         29.
   34.         38.         43.         48.         53.         58.
  63.         68.         74.         79.         84.         88.
  47.4766677  42.          ]]
```

3. Output Images



4. Challenges and Solutions

我認為這份作業最困難的地方在於理解整個任務概念，雖然只需完成merge.py檔案，但依舊需要去理解這個任務是怎麼完成的，還有merge函式會需要做什麼事，接受到的subset長什麼樣子，該對subset做什麼樣的處理，這些是我認為這份作業最需要花時間去理解的部分，雖然花了大部分時間去了解任務本身，但我認為這對於完成這份作業是有幫助的，可以讓我更加了解程式碼該如何撰寫。

程式碼的部分在尋找common_parts時很容易混淆，有時會搞不清楚是索引還是本身的值，且在決定該合併i的骨架還是j的骨架時也摸不著頭緒，或是合併出一個新的骨架，在多次和chatgpt協作後才了解可以將後者合併到前者並刪除後者，是個讓我比較不會混亂的作法。