Task 1

Assumption: Only focus on the read performance. Write performance and memory/storage is not a big concern.

- Hierarchy file is small
- Access pattern mostly likely supporting above operation
- Like write-once, read-many during the whole training and inferencing lifecycle

There are two relationships, i.e. **Part** and **Subcategory**

The hierarchy data is just a tree, with support graph-like operation

Operation	**Required mapping**
Find all siblings class of a class name	Child-to-parent, parent-to-child mapping
Find the parent class of a class name	Child-to-parent mapping
Find all ancestor classes of a class name	Child-to-parent mapping
Find if both class 1 and class 2 belong to the same ancestor class(es)	1. Get the root for two classes Child-to-parent mapping or node-to-leaf mapping 2. Check if root are the same

Parser for Open Image's Class Hierarchy

Assumption: Only focus on the read performance. Write performance and memory/storage is not a big concern.

- Hierarchy file is small
- Access pattern mostly likely supporting above operation
- Like write-once, read-many during the whole training and inferencing lifecycle

There are two proposed solutions for high read performance under python.

First Solution: Parse the JSON into bi-directional linked list. And compute the required result on-the-fire.

PRO: lower memory.

CONS: a bit slower the second one

Second Solution: Parse the JSON into bi-directional linked list. And built mapping caching layer for the required result directly.

Higher: Super fast in read performance

CONS: Taker more time in the parsing step (Write) and consuming more memory

Remark: Code written in lower-level language (e.g. C, rust) will be certainly more efficient if that is mission critical function.

First solution is implemented in the python notebook “task1\notebook\classParser.ipynb”

Task 2

Image Classification Problem ‘

The work is implemented in the python notebook “task2\notebook\PPE_classification.ipynb

Multi-label, and also multi-class (for mask) image classification problem

1. Annotation XML Parser
2. Train/Validation set the same , and overfitting the model to try
   --> **Over fit to see if transferred learning is working**.
   a. Helmet Model (Binary Classes)
      i. Try the Resnet-18 (without freezing),
         - train Loss: 0.2458 Acc: 0.9040
         - val Loss: 0.1827 Acc: 0.9336
      ii. ~~Try the Resnet-18 (with freezing), Train/Validation set the same.~~
         - ~~train Loss: 0.4761 Acc: 0.7673~~
         - ~~val Loss: 0.4241 Acc: 0.8036~~
         - ~~Best val Acc: 0.815290~~
   b. Mask Model (3 Classes)
      - Try the Resnet-18
         - train Loss: 0.4549 Acc: 0.8309 val Loss: 0.3101 Acc: 0.9018
         - Best val Acc: 0.910156
2. Train/Validation/Test Split - 0.7/0.2/0.1
   a. Helmet Model
      - train Loss: 0.2768 Acc: 0.8892
      - val Loss: 0.3615 Acc: 0.8380
      - Best val Acc: 0.837989
   b. Mask Model (Mask Model is overfitted, should be further tuned)
      - train Loss: 0.4585 Acc: 0.8429
      - val Loss: 0.6914 Acc: 0.7207
      - Best val Acc: 0.740223
3. Test and Evaluation
   - Accuracy, Recall and Precision
      - Mask Accuracy = 0.6871508379888268, Helmet Accuracy = 0.8547486033519553, Combined Accuracy = 0.5810055865921788
      - Mask Precision = 0.6875, Helmet Precision = 0.8582089552238806, Combined Precision = 0.75
      - Mask Recall = 0.46808510638297873, Helmet Recall = 0.9426229508196722, Combined Recall = 0.45
      - Mask F1 = 0.5569620253164557, Helmet F1 = 0.8984375000000001, Combined F1 = 0.5625000000000001
   - The combined metrics are to evaluate the stacked model (Helmet and Mask model).
   - For the evaluation, assume the scenario is a checking station before a working place, AUC can be used for this case. As usually the goal is to be functional (True positive Rate) while minimize the disturbance to user (False positive Rate).
     However, to use this evaluation metric, extra formulation is required to turn this problem into a binary classification, which has not been done in this work.
   - For the model diagnosis, each breakdown evaluation is with their purpose and certainly they should be kept for analysis

4. Setup naïve MLOps for temporary use
    a. Wandb/ClearML
5. Error and converge analysis
    a. Pattern lead to larger error
6. Potential Direction
    a. Upsampling
        i. Recrop a slightly deviated size apart from the annotated box in xml
        ii. Normal data augmentation
    b. Label shifting for Mask Model
        i. Combine two classes the invisible and no together (Class wrong is ignored)