YOLO

How YOLO Works:

Input: An image is input into the network.

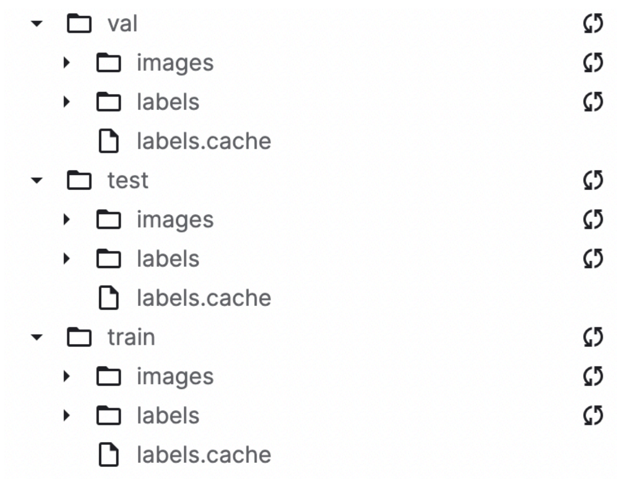
Network Processing: The image is divided into a grid. Each grid cell is responsible for predicting bounding boxes and their corresponding confidence scores for objects whose center falls within the cell. Alongside, each grid cell predicts class probabilities.

Output: For each bounding box, the model outputs a confidence score that reflects how confident the model is that the box contains an object and also how accurate it thinks the box is. This is combined with the class probabilities to give a final score for each class within each bounding box. The model outputs bounding boxes and class predictions for each grid cell.

Dataset Preparation

Fracatlas (https://www.kaggle.com/datasets/abdelazizfaramawy/fracatlas). There are 717 images with 922 instances of fractures. Each of the fracture instances has its own mask and bounding box annotated as part of the dataset.

The dataset is split into train, validation and test as per the splitup provided in the Fracatlas dataset. The images are then stored in a format structure as per yolo standards. The labels are generated from the coco annotations into the respective folders.
Dataset Folder Structure for YOLO -



```
val
    images
    labels
    labels.cache
test
    images
    labels
    labels.cache
train
    images
    labels
    labels.cache
```

Model Training

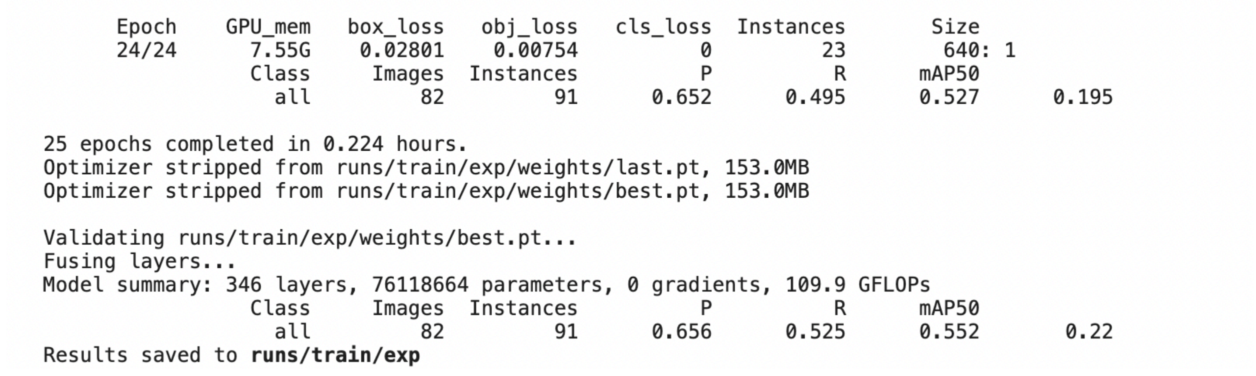
Pretrained model from the models available as part of yolov5 setup is used. The yolov5l6.pt file is a pre-trained model of the YOLOv5 family, specifically a larger variant with added depth and width for increased accuracy, along with an extended input resolution.

Features of YOLOv5l6

- **Increased Accuracy:** Compared to its smaller counterparts (s, m, x), the l model has more layers and parameters, allowing it to learn more complex features from the training data. This typically results in higher accuracy, especially in datasets with a wide variety of object sizes and complex scenes.
- **Extended Resolution:** The 6 in yolov5l6 suggests an adaptation for higher resolution inputs, which is particularly useful for detecting small objects in large scenes. It implies that the architecture is optimized to work with larger input sizes without a significant loss in processing speed or accuracy.
- **Application:** Given its design for larger-scale and higher resolution images, yolov5l6 is well-suited for applications requiring precise object detection over vast areas or detailed images, such as satellite image analysis, large-scale surveillance, or high-quality medical image processing.

Training Results

It was observed that the model required more number of epochs (25 epochs) to improve as the usecase is to detect smaller sections of fractures from an image which is different from the standard training done for YOLO models.



```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      24/24      7.55G    0.02801    0.00754          0         23        640: 1
                 Class     Images  Instances          P          R      mAP50
                   all         82         91      0.652      0.495      0.527      0.195

25 epochs completed in 0.224 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 153.0MB
Optimizer stripped from runs/train/exp/weights/best.pt, 153.0MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 346 layers, 76118664 parameters, 0 gradients, 109.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all         82         91      0.656      0.525      0.552       0.22
Results saved to runs/train/exp
```

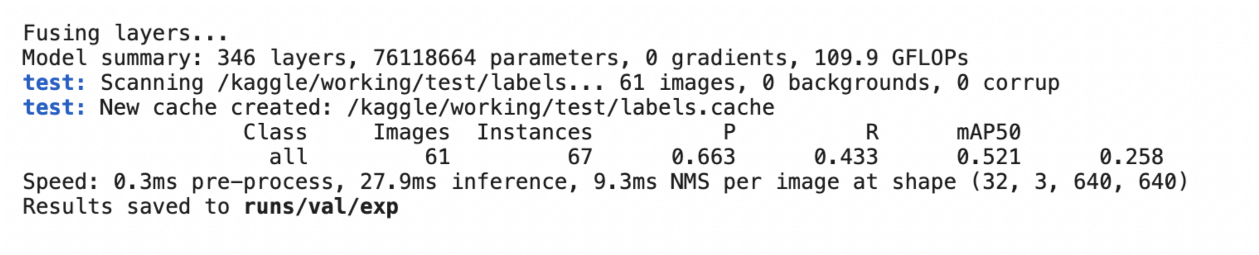
The final results from the YOLOv5 training session provide insights into the model's performance across various metrics at the last epoch (24/24). Here's a detailed explanation of the key metrics and outcomes:

- Box Loss (box_loss): The box loss measures how well the model predicts the bounding box coordinates for each detected object. A lower box loss indicates better performance. The final box loss is 0.03014, suggesting the model has learned to accurately predict the locations of objects.
- Object Loss (obj_loss): The object loss quantifies the model's ability to correctly identify the presence of objects within the predicted bounding boxes. The final object loss is 0.007678, which is relatively low, demonstrating the model's effectiveness in detecting objects.

- Class Loss (cls_loss): Since there's no class loss reported (0), it implies that this particular training session might have been focused on a single-class object detection or that class differentiation was not relevant for the model's objectives.
- Instances: The number of instances (objects) detected in the images processed in the last batch is 31. This metric helps in understanding the model's capacity to handle multiple objects within images.
- Size: The input image size for the model is 640 pixels, indicating the resolution at which the model processes images. This is a standard size for YOLOv5, balancing between speed and accuracy.
- Class Metrics (P, R, mAP50, mAP50-95):
    - Precision (P): The precision at the last epoch is 0.639, indicating that 63.9% of the objects the model predicted as present were indeed correctly identified. Precision is a key metric for evaluating the accuracy of the predictions.
    - Recall (R): The recall is 0.526, meaning the model correctly identified 52.6% of all actual objects. Recall measures the model's ability to detect all relevant cases in the dataset.
    - mAP50: The mean Average Precision at an Intersection over Union (IoU) threshold of 0.5 is 0.516. This metric demonstrates the model's accuracy in object detection at a specific IoU threshold, with higher values indicating better performance.
    - mAP50-95: This is an average mAP calculated over multiple IoU thresholds from 0.5 to 0.95 (in steps of 0.05). The final value is 0.196, offering a more comprehensive view of the model's performance across different levels of detection strictness.

These metrics collectively provide a comprehensive view of the model's performance. The relatively high precision (63.9%) and mAP50 (51.6%) suggest the model is quite accurate in detecting objects with a moderate level of recall (52.6%), indicating room for improvement in identifying all relevant objects within the images. The mAP50-95 value (19.6%) further underscores the model's effectiveness across various detection strictness levels, though it also highlights areas for potential enhancement, especially in more challenging detection scenarios.

Test Results -



```
Fusing layers...
Model summary: 346 layers, 76118664 parameters, 0 gradients, 109.9 GFLOPs
test: Scanning /kaggle/working/test/labels... 61 images, 0 backgrounds, 0 corrup
test: New cache created: /kaggle/working/test/labels.cache
                 Class     Images  Instances          P          R      mAP50
                   all         61         67      0.663      0.433      0.521      0.258
Speed: 0.3ms pre-process, 27.9ms inference, 9.3ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp
```

Overall, these test results suggest that while the model demonstrates a reasonable level of precision, its recall and mAP across stricter IoU thresholds indicate room for improvement. Enhancing recall without significantly sacrificing precision might involve further training, data augmentation, or model tuning to better capture the variety of objects within the dataset.

Images from the Test set after detection