**Tooth Detection using YOLOv8**

# 1. Confusion Matrix and Performance Metrics

The confusion matrix generated during validation illustrates the per-class performance of the model across all tooth categories. This matrix shows how accurately the model classified each tooth type.

Key metrics from evaluation on the test set:

- mAP@50: 0.946 (94.6%)

- mAP@50-95: 0.644 (64.4%)

- Precision: 0.908 (90.8%)

- Recall: 0.911 (91.1%)

Class-wise performance metrics are also available in the evaluation logs. These metrics confirm strong detection accuracy across molars, premolars, incisors, and canines.

# 

# 2. Summary of Approach

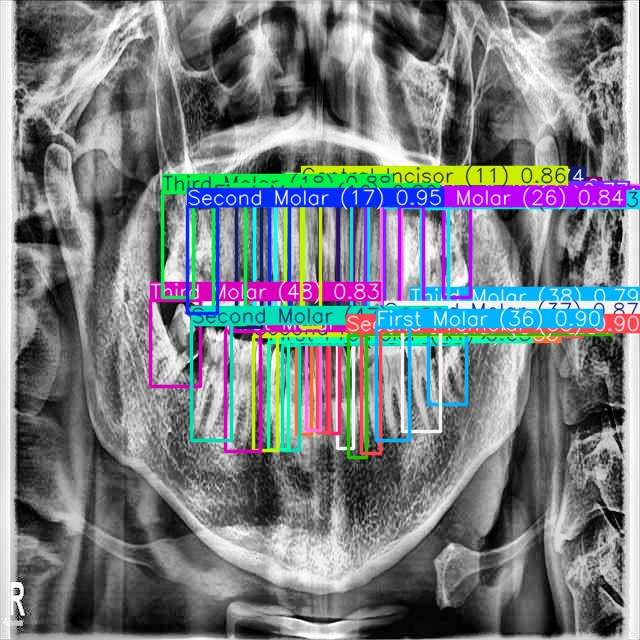
A YOLOv8-based object detection model is trained to detect and classify teeth using the FDI numbering system. The dataset was annotated in YOLO format, and training was performed with the Ultralytics YOLO framework.

The training process involved multiple epochs, and the best checkpoint (best.pt) was selected based on validation mAP@50. The model achieved strong results with 94.6% mAP@50 and over 90% precision and recall. This indicates the model is highly capable of detecting teeth accurately.

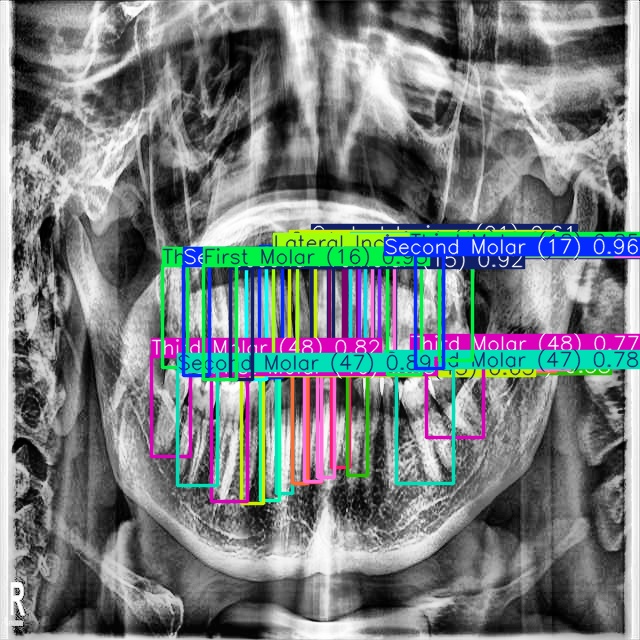
We evaluated the model on the test set and generated sample predictions. Both quantitative and qualitative results show the model generalizes well.

# 3. Sample Predictions

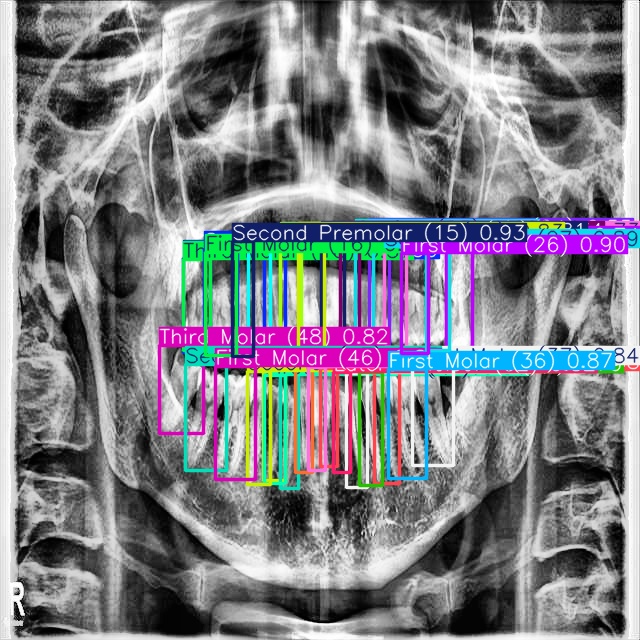
The following are example predictions generated by the trained model on test set images. The bounding boxes indicate the detected teeth, along with their labels and confidence scores.



Sample Prediction 1



Sample Prediction 2



Sample Prediction 3

# 4. Repository Structure

The repository is organized as follows:  
  
 - data.yaml: Dataset configuration file.  
 - train\_tooth\_detection.py: Training script for YOLOv8 model.  
 - model\_evaluation.py: Script to evaluate the trained model, generate metrics, and sample predictions.  
 - runs/: Folder automatically created by YOLO containing training and evaluation logs, weights, and plots.  
 - final\_sample\_predictions/: Contains annotated images with model predictions.  
 - README.md: Instructions for environment setup, training, and evaluation.

# 5. How to Run the Files

1. Install dependencies:

pip install ultralytics torch opencv-python matplotlib seaborn pyyaml

2. Train the model:

yolo detect train data=data.yaml model=yolov8s.pt epochs=100 imgsz=640

3. Evaluate the trained model:

python model\_evaluation.py

4. Run prediction on a new image:

yolo predict model=tooth\_detection\_runs/tooth\_detection\_exp/weights/best.pt source=path/to/image.jpg

# 6. Output Description

- The evaluation script prints key metrics such as mAP, precision, and recall.  
 - Confusion matrix and training plots (results.png) are generated automatically and stored in runs/.  
 - Sample predictions with bounding boxes and labels are saved to final\_sample\_predictions/.  
 - During inference, the model outputs annotated images along with detected class labels and confidence scores.

# 7. Prediction and Comparison with Ground Truth

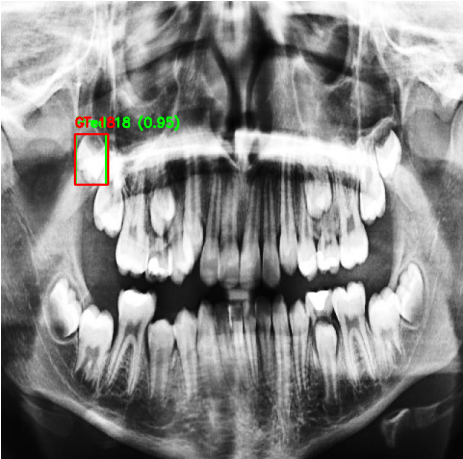
There is a script (`prediction.py`) that allows the user to input an FDI tooth number and compare the predicted bounding box with the actual ground truth bounding box from the dataset.  
  
This script loads the trained model (`best.pt`) and runs detection on a chosen image from the test set. It then overlays the predicted bounding box (green) and the ground truth bounding box (red) on the same image.

## How to Run

1. Update the following paths inside `prediction.py`:  
 - `BEST\_MODEL\_PATH`: Path to the trained `best.pt` model.  
 - `DATA\_YAML\_PATH`: Path to `data.yaml`.  
 - `SAMPLE\_IMAGE`: Path to a test image.  
 - `LABELS\_FOLDER`: Path to the corresponding labels folder.  
  
2. Run the script:  
 ```bash  
 python prediction.py  
 ```  
  
3. Enter the FDI tooth number (e.g., 11, 36, 48) when prompted.  
  
4. The script will display an image with:  
 - Predicted bounding box in GREEN (with confidence).  
 - Ground truth bounding box in RED (from labels).  
  
5. The comparison image will also be saved locally (e.g., `compare\_tooth\_11.jpg`).

## Example Output

Below is a description of the output:  
- Predicted tooth location is shown in GREEN.  
- Ground truth (from dataset labels) is shown in RED.  
- Both boxes appear on the same image for visual comparison.  
  
This helps in evaluating whether the model prediction aligns closely with the labeled ground truth.



The model has correctly predicted the tooth number 18

# 8. GitHub Repository Link

https://github.com/basu404/Tooth-Detection-Model/