# YOLO-Based Object Detection for Tooth Analysis

## Executive Summary
This project aimed to enhance dental diagnostics through the application of deep learning, specifically using the YOLOv8 and YOLOv11 object detection models. We developed a system that detects cavities and plaque in intraoral dental images. Our team created a full pipeline from data preprocessing to model training, testing, and packaging for deployment. This final deliverable includes a polished and documented version of our work ready for reproduction or future extension by clinical researchers or developers.

## Project Goals
- Develop a machine learning pipeline capable of detecting dental anomalies from intraoral images.
- Train and evaluate a YOLOv8 and YOLOv11 model to identify cavities and plaque.
- Create a lightweight, reproducible system deployable by dental professionals or researchers.
- Deliver clear documentation, user instructions, and a demo for end users.

## Project Methodology
- **Data Collection**: Images were sourced from Kaggle. The dataset included intraoral dental images used to train the model for caries detection.
- **Data Cleanng**: Python libraries such as NumPy and Pandas were used to clean the data. This involved removing duplicate and blurry images to ensure high-quality input.
- **splitting Data**: The dataset was divided into training, testing, and validation sets using the stratified sampling method to maintain class distribution across all subsets. The stratified sampling method was specifically used for image splitting to ensure that each class was fairly represented.
- **Data Augmentaion**: Various augmentation techniques, such as flipping, rotating, resizing and cropping, were applied to the training images to increase diversity and help the model generalize better.
- **Image Labelling**: Images in the training and validation sets were labeled using LabelImg and MakeSense.AI. This labeling helped the model learn to differentiate between caries, healthy teeth, and background.
- **Model Selection**: Initial experiments were conducted with YOLOv8, which showed limitations due to small dataset size. The project later shifted to YOLOv11, which was better suited for larger datasets and achieved higher accuracy.
- **Model Training**: Models were trained using Ultralytics YOLO framework. Both non-augmented and augmented versions of the dataset were used to compare performance. Evaluation was done using F1 Score, Precision, Recall, and mAP0.5.

## Results / Findings
- Successfully implemented a YOLOv8 and YOLOv11 object detection model to identify two target classes: cavities and plaque.

```markdown
- Identified early overfitting and applied augmentation techniques to control the risk of overfitting.
### YOLOv8
- The model shows moderate ability to detect carious lesions and struggled to detect healthy teeth, likely due to class
- imbalance or insufficient examples.
- High-confidence detections (>0.80) were observed for clear and well-lit caries, however, Dense regions with
overlapping caries - posed a challenge, often resulting in cluttered bounding boxes with low cofidence score.
- Augmented data improved model robustness but did not significantly increase AP.
- Data quality and annotation consistency proved more important than just dataset size.
### YOLOv11
- Achieved stable validation accuracy (~85%) after 100 epochs of training.
- 60% of cavity and plaque cases were correctly classified and remaining 40% were primarily misclassified as
background, indicating room for improvement in edge cases and finer object distinctions.
- The model generalized well despite the absence of augmentation, showing promise for broader deployment.
- Deployed model to test on unseen samples, with real-time inference capabilities.
- Final model and documentation uploaded to GitHub for public access.
### Model Comparison
- YOLOv11 outperformed YOLOv8 due to:
- Larger and more balanced dataset
- Better F1 and mAP scores
- Superior detection of both anterior and posterior dental conditions
- On the other hand, YOLOv8, although lightweight and efficient, was limited by:
- Dataset size and quality
- Inability to learn healthy class patterns

**Key Outcomes:**
* Labeled dataset prepared and visually validated.
* YOLOv8 and YOLOv11 trained for dental object detection.
* Binary CNN model also explored for baseline classification.
* Project packaged for reproducibility and deployment.
* Presentation and demo materials completed.

## Demo
https://unomaha.yuja.com/V/Video?v=13022187&node=56381952&a=89808787

## Requirements:


We have established and documented the required environments, including:
```

* YOLOv8 (AI object detection).

* Kaggle Datasets (Training images).
* Python Libraries (NumPy, OpenCV, TensorFlow, Pandas, matplotlib).
* GitHub (Version control).
* Other Libraries (Shutil, OS, sklearn, seaborn, YOLO).
* Installing necessary dependencies for image processing and augmentation.
* Divide the images into training, validation, and test sets using stratified sampling.
* Used LabelImg to label objects(caries) in the images.
* Setting up repositories and ensuring proper access for all team members.
* Added liscence GNU General Public License v3.0.
* Using platforms for programming - Visual studio, Jupiter notebook.

YOLOv8 Quickstart

See below for quickstart installation and usage examples. For comprehensive guidance on training, validation, prediction, and deployment, refer to the official full Ultralytics Docs.

YOLOv11 Quickstart

## How to Run the Model

1. Prepare the Dataset

```
python scripts/prepare_data.py
```

2. Train the model

```
python script/train.py
```

Results will be saved across multiple training runs inside the runs/detect/ directory.

## Install

Install the ultralytics package in a Python ≥ 3.8 environment with PyTorch ≥ 1.8:

For alternative installation methods (Conda, Docker, source build), see the Ultralytics Quickstart Guide.

## Usage

Command Line Interface (CLI) Run predictions, training, validation, and exports directly from your terminal:

## Predict with a pretrained YOLOv8n model

`yolo predict model=yolov8n.pt` For additional CLI examples, check the YOLO CLI Docs.

### Python API

Embed YOLOv8 into your Python projects with the same flexibility as the CLI:

```
from ultralytics import YOLO
```

### Load a pretrained YOLOv8n model

```
model = YOLO("yolov8n.pt")
```

```

### Train on your dataset (e.g., coco8.yaml) for 50 epochs at 640px
model.train(data="coco8.yaml", epochs=50, imgsz=640, device="0")

### Validate on the validation split
model.val()

### Run inference on an image
results = model("path/to/image.jpg")
results[0].show()

### Export to ONNX
model.export(format="onnx")

## The yolo11 model also has similar installation steps as yolo8.
### NumPy
here are the quick start installations steps and usage instructions for NumPy https://numpy.org/install/
Install
pip install numpy
Usage
To perform the mathematical operations
Open CV
here are the quick start installations steps and usage instructions for Open CV (https://pypi.org/project/opencv-
python/#installation-and-usage )
### Open CV
here are the quick start installations steps and usage instructions for Open CV (https://pypi.org/project/opencv-
python/#installation-and-usage )
### Install
pip install opencv
Usage
Used for computer vision and processing
### Tensor Flow
here are the quick start installations steps and usage instructions for Tensor Flow(https://pypi.org/project/tensorflow/ )
### Install
pip install tensorflow
### usage
Used for high performance numerical computation.
```

```
### Pandas
here are the quick start installations steps and usage instructions for
Pandas(https://pypi.org/project/pandas/#documentation )
### Install
pip install pandas
### Usage
Used for data analysis and manipulation.
### Matplotlib
here are the quick start installations steps and usage instructions for
matplotlibs(https://matplotlib.org/stable/install/index.html )
### Install
pip install matplotlib
### Usage
Used for data visualizations.
### sklearn
here are the quick start installations steps and usage instructions for sklearn(https://pypi.org/project/scikit-learn/ )
### Install
pip install scikit-learn
### Usage
To provide the comprehensive tool kit for machine learning
### seaborn
here are the quick start installations steps and usage instructions for seaborn(https://pypi.org/project/seaborn/ )
### Install
pip install seaborn
### Usage
Visualization library for statistical graphics plotting in python
### LabelImg
here are the quick start installations steps and usage instructions for LabelImg(https://pypi.org/project/labelImg/ )
### Install
pip3 install labelImg labelImg labelImg [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
### Usage
Used for annotaions of images

## ⚙️ Installation & Setup Guide

This guide outlines how to set up the environment to train and test the YOLOv11-based dental cavity detection
model.
```

~~~markdown
###  1. Clone the Repository

```bash
git clone https://github.com/daunkim18/Capstone-Project.git
cd Capstone-Project/milestone3
```

### 2. Create & Activate Python Environment

> Recommended Python version: **3.10** or **3.11**

```bash
# Create a virtual environment
python -m venv venv

# Activate it
source venv/bin/activate        # On macOS/Linux
venv\Scripts\activate         # On Windows
```

### 3. Install Required Packages

```bash
pip install --upgrade pip
pip install -r requirements.txt
```

If you don’t have a requirements file, you can manually install:

```bash
pip install ultralytics opencv-python pandas matplotlib
```

---

### 4. Folder Structure
~~~

~~~
```
Capstone-Project/
 └─── milestone3/
     ├─── data/
     │    ├─── images/
     │    │    ├─── train/
     │    │    ├─── val/
     │    │    └─── test/
     │    ├─── labels/
     │    │    ├─── train/
     │    │    ├─── val/
     │    │    └─── test/
     │    └─── data.yaml
     ├─── scripts/
     │    ├─── prepare_data.py
     │    ├─── train.py
     │    └─── demo_yolo11.py
     ├─── runs/
     └─── yolov8n.pt
     └─── yolov11n.pt
```

---

## Model Training

### 🔷 YOLOv8

```bash
python scripts/prepare_data.py
python scripts/train.py
```

### 🔷 YOLOv11

```bash
~~~

~~~markdown
python scripts/train.py --data configs/data.yaml --epochs 100 --batch 8 --imgsz 640
```

Results will appear under `runs/detect`

---

## 🎯 Inference Demo

Ensure these files exist:
- `data/images/test/demo.jpg`
- `data/labels/test/demo.txt`

Then run:
```bash
python scripts/demo_yolo11.py
```

Output: `demo_output.jpg` with detected bounding boxes.

---

## YOLO CLI Prediction (Optional)

```bash
yolo task=detect mode=predict model=runs/detect/train14/weights/best.pt source=data/images/test/
```


---

## Notes
- Label format is YOLO (.txt) with class ID and normalized coordinates.
- Make sure all images have corresponding label files in `labels/`.
- Demo script uses OpenCV to visualize bounding boxes.

---
~~~

~~~~
Then:


```bash
python scripts/demo_yolo11.py
```


You will get a visual output in `demo_output.jpg`.



### Getting Started
#### To Preprocess Images:
1. Place labeled images into the data/raw/ folder.
2. Run:
   python src/preprocess_images.py
#### To Train YOLO
1. Ensure your dataset is in YOLO format (images/train, images/val, and labels/).
2. Run:
   yolo task=detect mode=train model=yolov8n.pt data=dental.yaml epochs=20 imgsz=640
   yolo task=detect mode=train model=yolov11n.pt data=dental.yaml epochs=20 imgsz=640
#### To Make Predictions
1. Place test images into a folder (data/test/).
2. Run:
   yolo task=detect mode=predict model=runs/detect/train/weights/best.pt source=data/test/
## Final report Summery:
All intraoral images are present in the milestone2/stratified_sampling/
~~~~