Comprehensive Analysis of Object Detection and Image Segmentation on Dental X-ray Images Using YOLOv8 and Advanced Deep Learning Models

Karthik Ragulan

April 15, 2025

Abstract

This study explores deep learning methods for automated analysis of dental X-rays. YOLOv8 and segmentation models were used to detect teeth and isolate caries. Data augmentation and preprocessing addressed challenges like class imbalance and X-ray variability. Combining object detection with segmentation improved diagnostic accuracy. Key methodologies, results, and future work are discussed.

Contents

1 Introduction

Dental radiographs play a crucial role in identifying dental pathologies such as caries, periodontal diseases, and structural abnormalities. However, manual analysis of these images is often time-consuming and prone to subjectivity. Recent advancements in deep learning have enabled automated analysis with high accuracy, offering significant potential for clinical applications.

This study focuses on two key tasks:

1. Object detection to identify regions of interest (cavities) in dental X-rays.
2. Image segmentation to isolate caries regions within the detected areas.

We employed YOLOv8 for object detection due to its state-of-the-art performance in real-time applications. For image segmentation, models such as U-Net,Resnet-32 based architectures were evaluated. The challenges faced during model training and testing are discussed in detail, along with the strategies employed to overcome them.

2 Methodology

The methodology is divided into two main tasks: object detection and image segmentation. Each task involves specific models and preprocessing techniques.

2.1 Dataset Description

The dataset consisted of annotated dental X-ray images.The annotations included bounding boxes for object detection (cavities) and pixel-level masks for caries segmentation.

Key characteristics of the dataset:

- Total images: 300
- Classes: cavities
- Segmentation masks: Binary masks for caries regions

Data preprocessing steps included resizing images to a uniform size, normalizing pixel values to [0,1].

2.2 Object Detection

Object detection was performed using YOLOv8 along with Faster R-CNN for comparison.

2.2.1 YOLOv8 Architecture

YOLOv8 employs a unified architecture with three main components:

- **Backbone:** CSPDarknet for feature extraction.
- **Neck:** PANet for feature aggregation.
- **Head:** Predicts bounding boxes, class probabilities, and confidence scores.

The model was trained using the following hyperparameters:

- Optimizer: AdamW
- Epochs: 500
- Loss function: Binary cross-entropy for classification and IoU loss for bounding box regression.

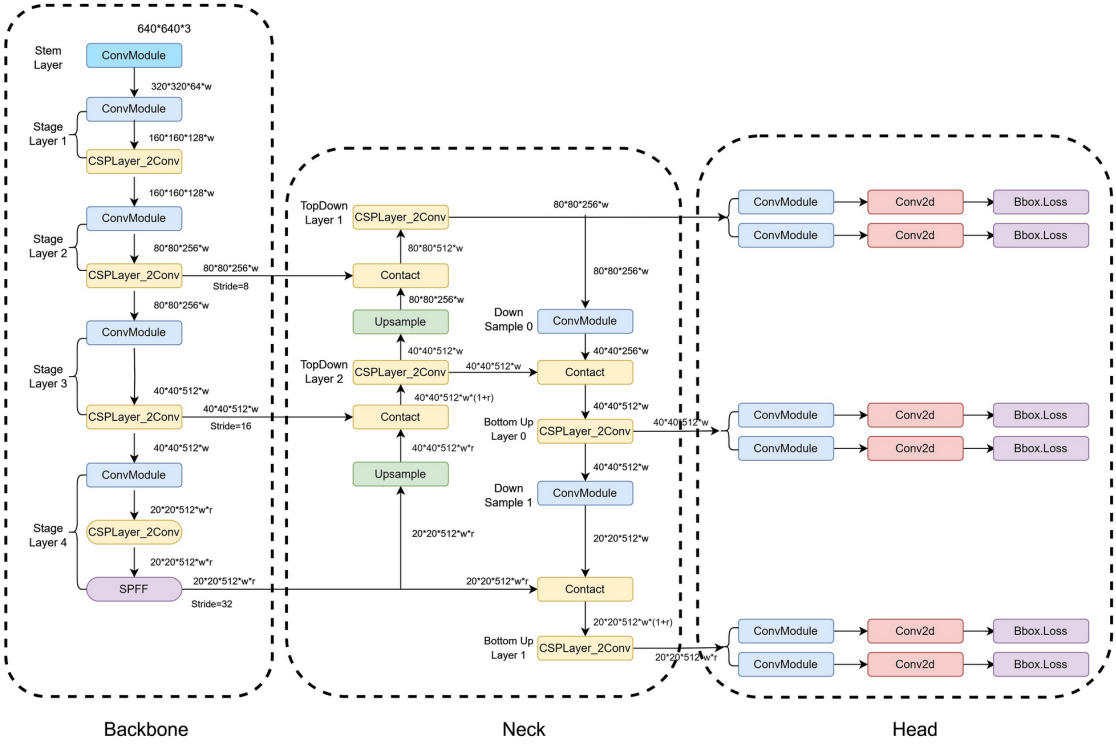

Figure 1: Overview of Yolov8 Architecture Used for Objedt Detection.

2.3 Image Segmentation

For caries segmentation, we evaluated multiple models:

1. U-Net: A fully convolutional network designed for biomedical image segmentation.
2. DeepLabV3+: We used the `smp.DeepLabV3Plus` model with two different encoders—ResNet and MiT-B5 —for image segmentation tasks.

Each model was trained on binary masks representing caries regions. The Dice coefficient is the evaluation metric.

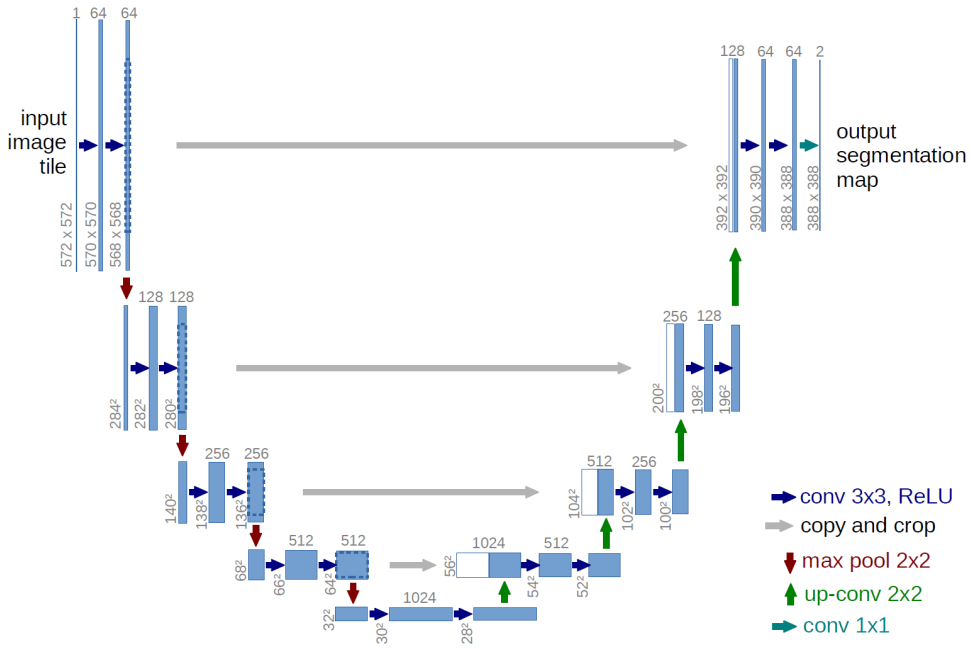

Figure 2: Overview of U-Net Architecture Used for Image Segmentation.

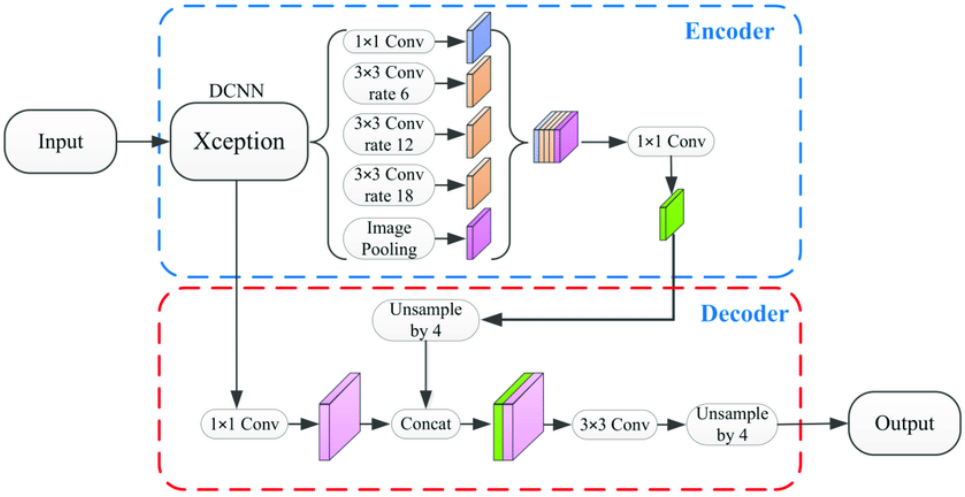

Figure 3: Overview of DeepLabV3+ Architecture Used for Image Segmentation.

Results indicated that DeeplabV2Plus model with MitB2 encoder outperformed other models with an Dice Score of 0.78 on the training data, and 0.45 Dice Score on validation data.

3 Challenges Faced During Implementation

Several challenges were encountered during this study:

1. **Data Imbalance and Insufficiency**: The dataset provided had very few samples, especially for caries cases, making it difficult to train a robust model.

   Solution: To better train the model better additional data were sourced from the internet to augment the dataset and improve model performance.

2. **Overfitting During Training**: Some models overfit due to limited training data.
   Solution: Employed dropout layers and early stopping during training.

3. **Annotation Errors**: The bounding boxes (annotations) provided in the dataset covered the caries in the image and also a vast region around it.
   Solution: Sourced dataset that had similar X-Ray images provided in the dataset , but with more tightly bound boxes around the cavities.

4 Object Detection

4.1 YOLOv8 Performance on Cavity-Only Dataset

In this project, YOLOv8 was trained exclusively on a dataset containing only images of dental cavities, with each image annotated using bounding boxes that enclosed the cavity regions. This focused approach was chosen to help the model learn the intricate visual patterns associated specifically with caries, without being distracted by healthy tooth structures.

4.2 Limitations and Future Directions

- **Lack of Negative Samples:** Since the dataset only contained cavity images, the model was not exposed to healthy teeth. This prevents it from distinguishing between healthy and diseased regions, and limits its real-world diagnostic utility.

- **Binary Generalization Problem:** The current model functions more as a "cavity localizer" than a classifier. Incorporating healthy images in the future would enable binary classification (cavity vs. no cavity), greatly enhancing clinical relevance.
- **Small Dataset Size:** Although tightly annotated, the number of samples remains limited. Expanding the dataset with more diverse cavity presentations would further improve robustness.

In summary, YOLOv8 demonstrates strong potential for automated cavity detection when trained on curated, cavity-only datasets. For deployment in real clinical scenarios, future efforts will involve expanding to include healthy cases and enabling binary decision-making for complete diagnostic support.

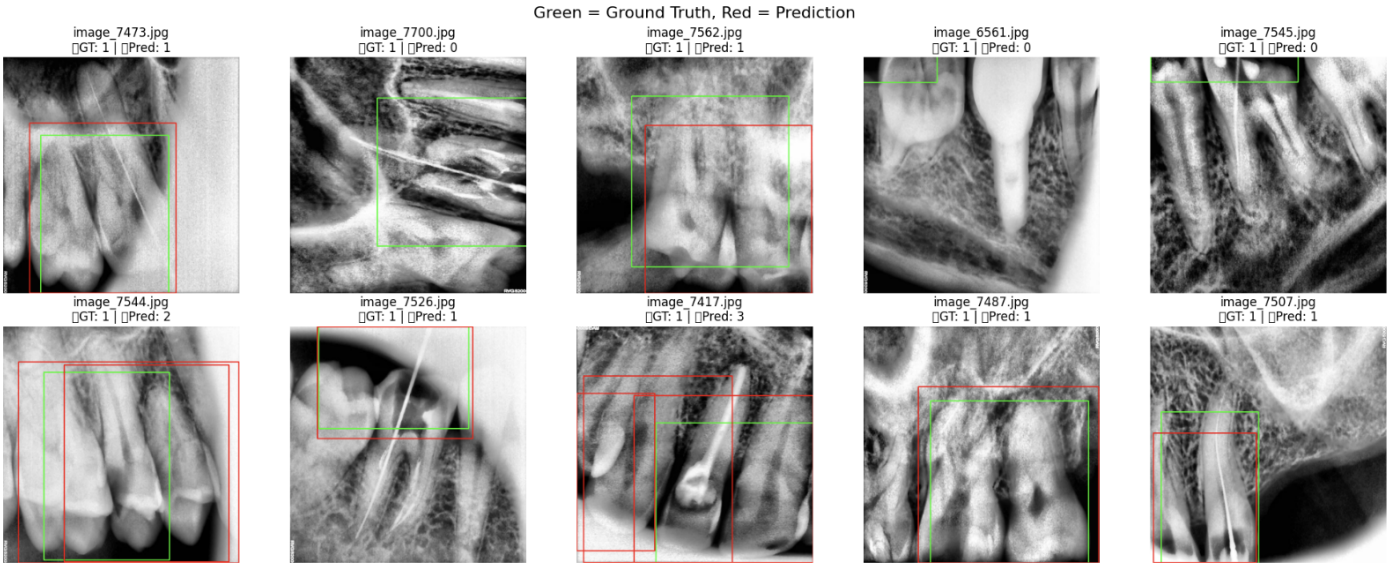

Figure 4: Predictions of Yolov8 on the original dataset.

4.3 Improving the Detection

One of the primary challenges we encountered during model training was the poor quality of bounding box annotations in the original dataset. Many of the provided boxes were overly large, often encompassing the entire tooth or significant surrounding regions rather than just the cavity. This introduced noise into the learning process, as the model struggled to distinguish between relevant and irrelevant features within the large boxed area.

To address this, we undertook a data enhancement process by sourcing additional high-quality images from the internet. These images specifically highlighted dental cavities and came with tight, manually annotated bounding boxes that focused precisely on the affected region. We prioritized selecting samples with varying cavity sizes, angles, lighting conditions, and tooth types to improve model robustness.

After incorporating these curated images, we observed a significant improvement in detection accuracy. The model began predicting more tightly bound boxes that aligned well with the actual cavity locations. Additionally, using these refined annotations helped reduce false positives on healthy teeth and improved the model's focus on clinically relevant regions.

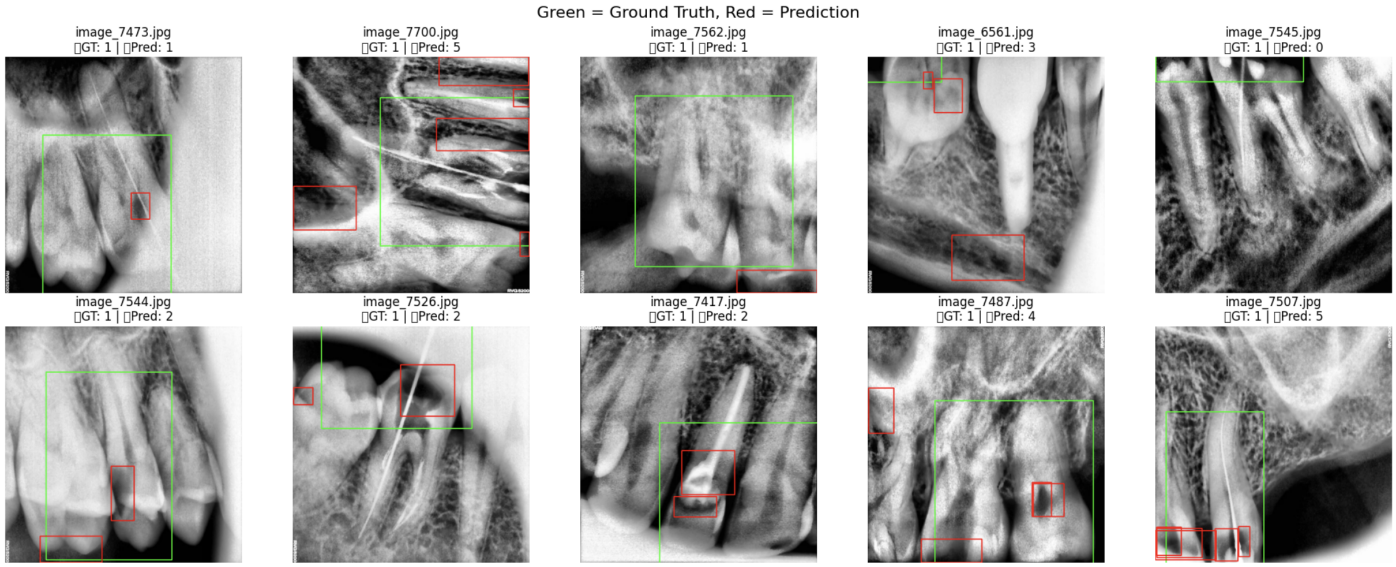

Figure 5: Predictions of Yolov8 on the original dataset while trianed on dataset with more tightly fit bounding boxes.

Figure 7 clearly illustrates the improvement in annotation quality. The tighter boxes provide more relevant features for the model to learn from, resulting in higher prediction precision.

In conclusion, combining better annotations with a more robust detection architecture like YOLOv8 significantly improved our caries detection pipeline. This dual strategy—model comparison and data enhancement—was key to achieving reliable results suitable for real-world application.

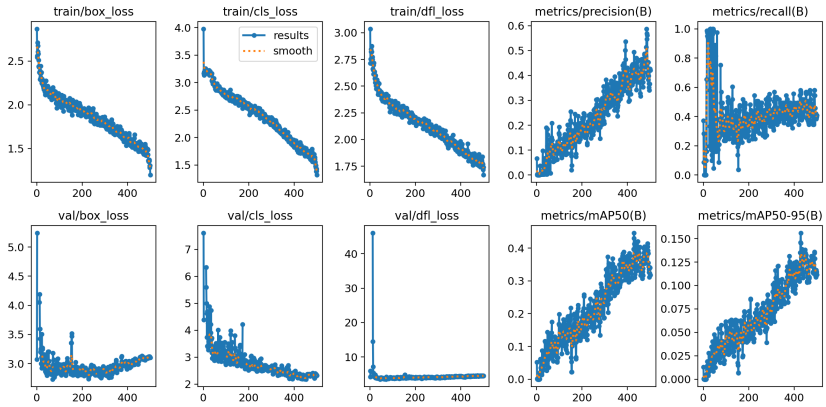

Figure 6: Metrics of YoloV8 on while trained on the original data.

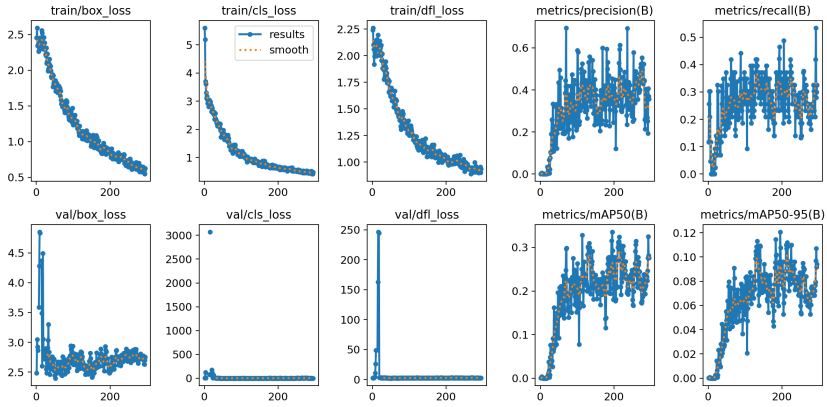

Figure 7: Metrics of YoloV8 on while trained on the data with tighter bounding boxes on the cavity.

5 Image Segmentation

Image segmentation is the process of partitioning an image into its constituent regions and is a key task for computer vision applications. Recent advances in deep learning have led to the development of architectures such as UNet, which effectively preserve spatial details while capturing high-level context. In this report, a UNet architecture is utilized for binary segmentation where the network predicts a mask corresponding to the region of interest.

5.1 Data Collection and Preprocessing

The dataset used consists of images and their corresponding ground truth masks. The masks are binary, with foreground objects denoted by white pixels and background pixels by black.

- **Image directory:** Contains raw images for training.
- **Mask directory:** Contains ground truth segmentation masks.

5.2 Preprocessing and Augmentation

The images and masks undergo several preprocessing steps:

- **Loading:** Images and masks are loaded using Python's `PIL` library.
- **Binarization:** The masks are binarized, where non-zero pixels are considered the foreground, and zero pixels are the background.
- **Resizing:** All images are resized to 256x256 to maintain a consistent input size for the model.
- **Augmentation:** We applied several augmentations to increase model robustness, including:
  - Horizontal flipping
  - Random rotation
  - Elastic transformations
  - Random brightness and contrast adjustments
- **Random Horizontal Flip:** Applied with a probability of 0.5, this augmentation mirrors the image along the vertical axis. Since objects of interest (e.g., organs, crops, roads) can appear symmetrically, this technique allows the model to learn invariance to horizontal orientation.
- **Random Rotation:** Images were rotated randomly within a range (e.g., $[-15°, +15°]$). This prevents the model from becoming biased to specific orientations present in the training set and promotes rotational invariance.
- **Random Crop and Resize:** Cropping was used to randomly focus on subregions of an image. Subsequently, images were resized to a fixed shape (e.g., 256×256). This not only standardizes input size but also helps the model learn to generalize across different spatial contexts.
- **Color Jittering:** This includes random adjustments in brightness, contrast, saturation, and hue. These augmentations simulate variations in lighting conditions and color distribution, allowing the model to be robust under different illumination settings.
- **Normalization:** Input images were normalized using the mean and standard deviation of the ImageNet dataset (`mean = [0.485, 0.456, 0.406]`, `std = [0.229, 0.224, 0.225]`). This ensures that pixel values are centered around zero and scaled appropriately, leading to faster convergence and more stable gradients during training.

The transformations are applied using the `albumentations` library, which is fast and efficient.

5.3 Model Architecture

For the segmentation task, we used the DeepLabV3+ model, which is based on the encoder-decoder architecture. It employs atrous (dilated) convolutions to capture contextual information at multiple scales.

DeepLabV3+ was chosen for its high performance in semantic segmentation tasks. We used the following components:

- **Encoder:** MIT-B5 (EfficientNet-based) encoder, which has been proven to be powerful for image feature extraction.
- **Decoder:** A decoding block that refines the feature map.
- **Activation:** No activation function was applied in the final output layer, as this is handled by the loss function.

Model Implementation: The model was implemented using the `segmentation_models_pytorch` library, which provides a ready-to-use DeepLabV3+ implementation with pre-trained weights from ImageNet.

The final loss function is a weighted sum of Dice and BCE losses:

$$\text{Loss} = 0.6 \times \text{DiceLoss} + 0.4 \times \text{BCEWithLogitsLoss}$$

5.4 Training Methodology

Once the data is augmented and loaded, a robust training loop is implemented to ensure convergence and prevent overfitting. Below is a breakdown of each component of the training process.

Data Loading

Two `DataLoader` objects were used for training and validation sets. Important optimizations include:

- `shuffle=True`: Ensures batches contain a diverse mixture of samples, helping avoid local minima.
- `pin_memory=True` and `non_blocking=True`: These enable faster data transfer to the GPU, reducing training time.

Loss Function

A composite **combined loss** was used, typically consisting of:

- **Binary Cross Entropy (BCE) Loss:** Penalizes per-pixel prediction error and helps in classifying each pixel independently.
- **Dice Loss:** Measures the overlap between the predicted and ground truth masks. Particularly effective for imbalanced segmentation problems, as it directly optimizes the metric of interest.

The combined loss leverages both pixel-wise accuracy and region-level overlap to guide the learning process.

Optimizer and Scheduler

An optimizer (e.g., Adam or SGD) was used for gradient descent. To improve convergence:

- A learning rate scheduler was applied to reduce the learning rate as training progressed, allowing the model to take larger steps initially and fine-tune with smaller steps later.

Gradient Clipping

To prevent the exploding gradient problem, `torch.nn.utils.clip_grad_norm_` was used with a maximum norm of 1.0. This keeps gradient values within a manageable range, especially helpful in deep architectures.

Early Stopping

To avoid overfitting and reduce training time, an early stopping mechanism was implemented:

- If the validation Dice score did not improve for 5 consecutive epochs, training was halted.
- This ensures the model retains its best generalization capability without unnecessary training on noise.

Model Checkpointing

The model with the highest validation Dice score was saved using:

```
torch.save(model.state_dict(), "best_deeplabv3plus_improved.pth")
```

After training, the best model was reloaded to perform inference and visualization.

5.5 Training and Validation Dice Scores

Below are the Dice scores for both the training and validation sets over 20 epochs:

Epoch	Train Dice Score	Validation Dice Score
1	0.7105	0.5982
2	0.7055	0.6004
3	0.7064	0.5980
4	0.7132	0.5956
5	0.7219	0.5864
6	0.7078	0.5792
7	0.7090	0.5855

Table 1: Dice Scores for Training and Validation (Epochs 1–7)

6 Evaluation and Visualization

After training, the model is set to evaluation mode. A helper function gathers a random selection of images from a validation set and:

- Computes the predictions using the trained model.
- Applies a sigmoid activation followed by a threshold of 0.5 to obtain a binary segmentation mask.
- Displays a grid that compares the original image, the ground truth, and the predicted mask.

This process helps qualitatively assess the performance of the segmentation model.

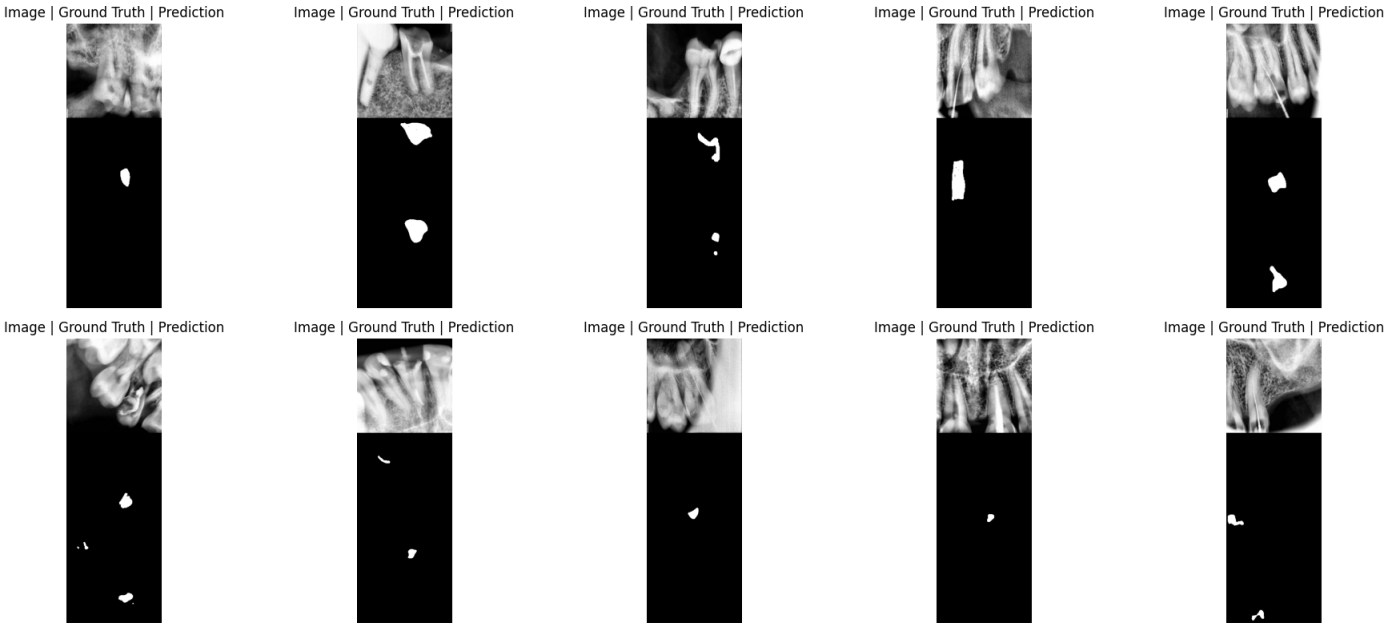

Figure 8: Cavity Segmentation using Vanilla Unet without any Augmentations.

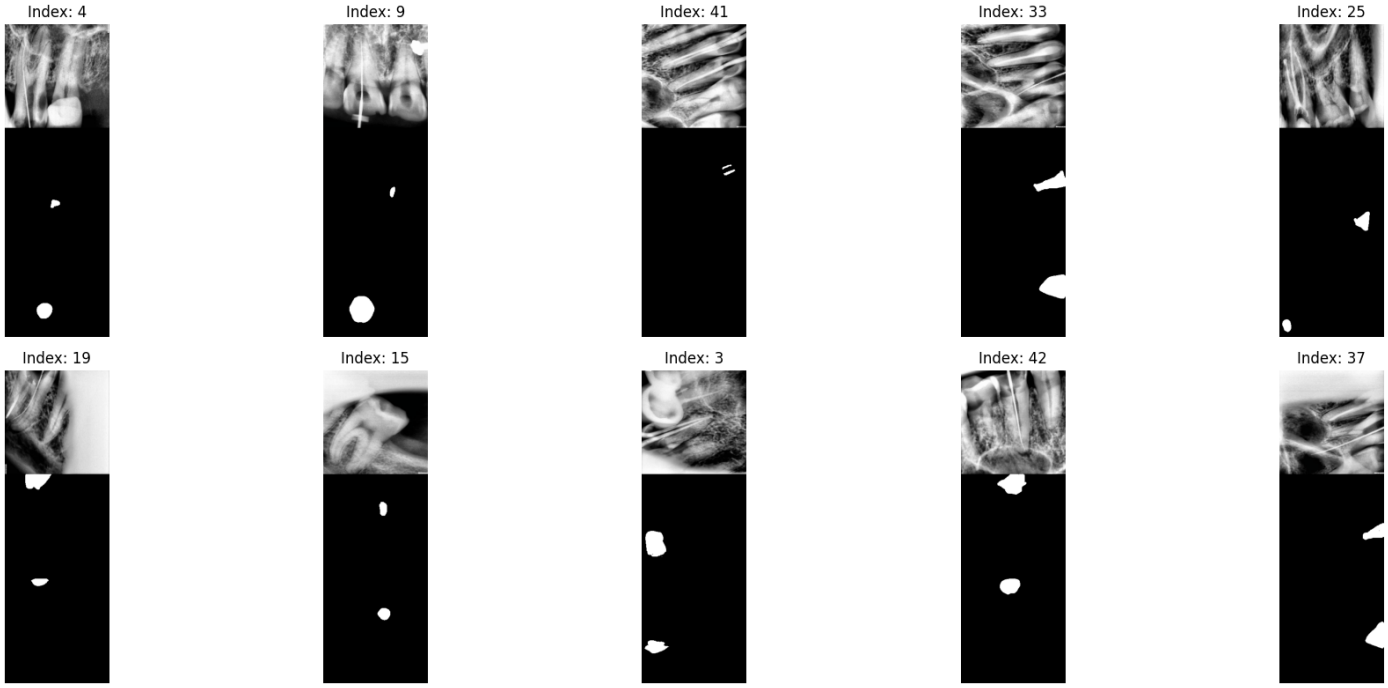

Figure 9: Cavity Segmentation using DeepLabV3+ with a resnet encoder and Augmentations.

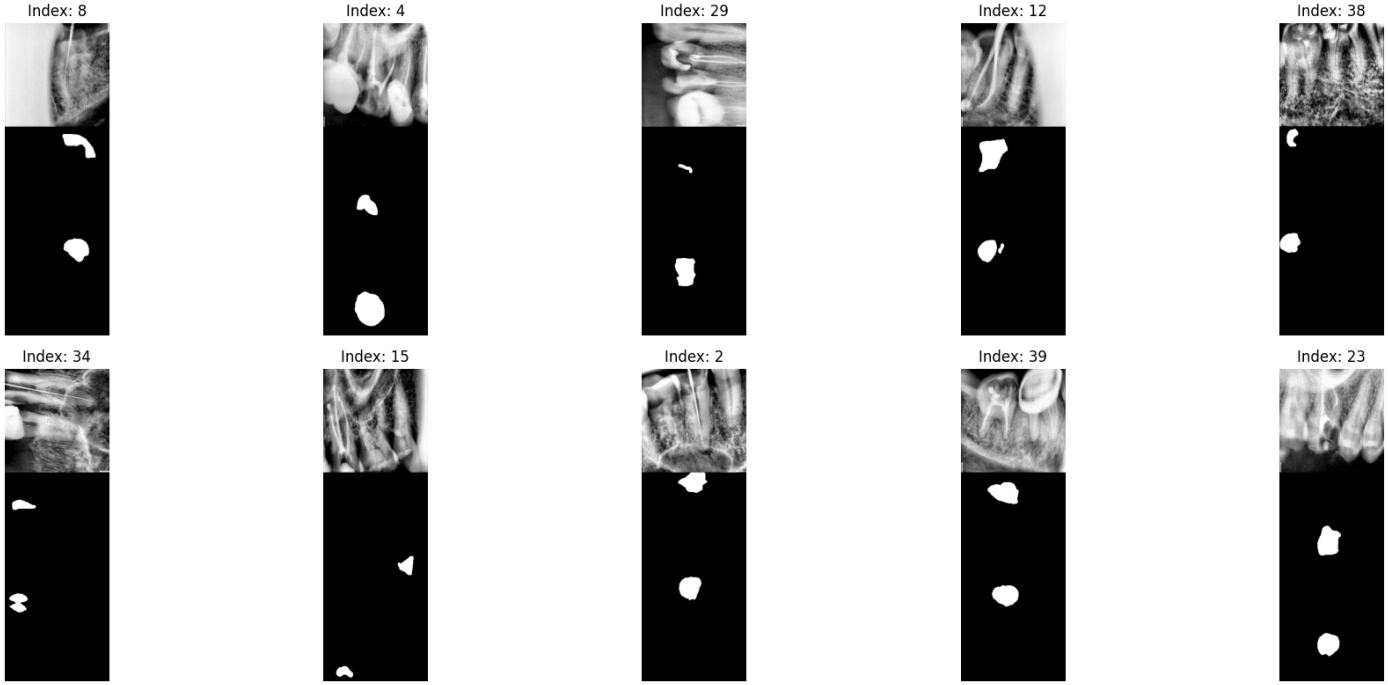

Figure 10: Cavity Segmentation using DeepLabV3+ with a MitB5 encoder and Augmentations.

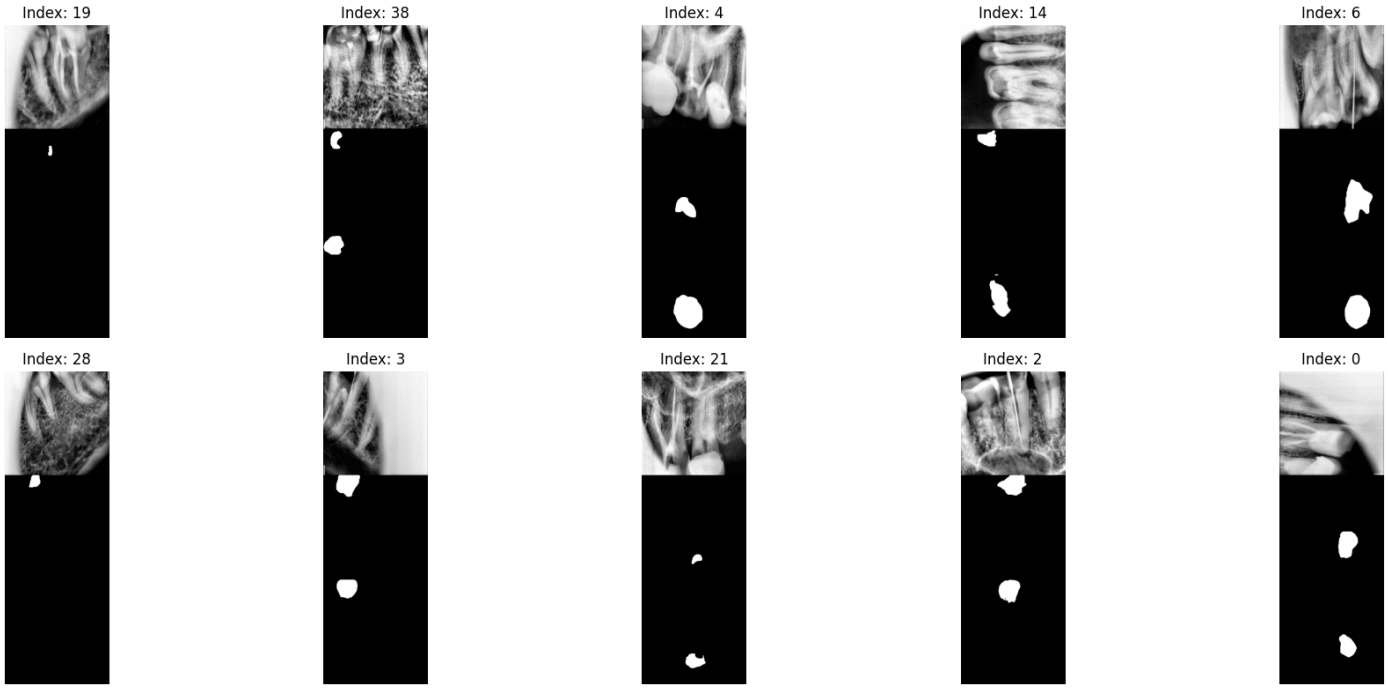

Figure 11: Cavity Segmentation using DeepLabV3+ with a MitB5 encoder and Augmentations and Early Stopping.

APPENDIX

While our current framework employs YOLO, and advanced preprocessing techniques with DeeplabV3+ and MitB5 we believe that further improvements can be achieved by exploring the following avenues:

- **Advanced Object Detection Frameworks:** Future work could investigate the use of transformer-based models such as **DETR** (DEtection TRansformer) and **DINO** (DETR with Improved Training Objectives). These frameworks leverage self-attention mechanisms to capture global context, which has been shown to improve robustness and accuracy in detecting objects within complex images.

- **Enhanced Data Representations via Tooth Contours:** In addition to using standard segmentation masks, incorporating more detailed annotations—specifically, the **contours of the teeth**—could improve cavity segmentation. By leveraging the geometric structure of dental contours, models can better delineate cavity boundaries, leading to a reduction in false positives and improved segmentation outcomes.
- **State-of-the-Art Segmentation Models:** Recent developments in segmentation, such as the **YOLOv8Seg** model, have shown promising results in terms of both speed and accuracy. Integrating these cutting-edge models into the pipeline may further enhance segmentation performance, particularly when dealing with fine-grained features and complex structures.

Conclusion

The report presents a comprehensive methodology for semantic image segmentation. By leveraging a UNet architecture with a robust pre-trained encoder and using Dice Loss optimized by the Adam optimizer, the approach efficiently learns to segment images. Both training and validation pipelines are well designed using Python libraries such as `torchvision` and `albumentations`, and visualization techniques validate the prediction quality. This method is typical of contemporary deep learning solutions in computer vision for tasks such as medical imaging, remote sensing, and autonomous driving applications.

References

[1] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.

[2] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241). Springer.

[3] Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, 801–818.

[4] Loshchilov, I., Hutter, F. (2019). Decoupled Weight Decay Regularization. In *International Conference on Learning Representations* (ICLR).

[5] Gonzalez, R. C., Woods, R. E. (2002). Digital Image Processing (2nd Edition). Prentice-Hall.

[6] Shorten, C., Khoshgoftaar, T.M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 60.