



Faculty of Computer Science and Information Technology

***CAVITY DETECTION IN DENTAL IMAGES USING DEEP
LEARNING***

Wong Guan Ting

Bachelor of Computer Science with Honours (Computational Science)
2025

CAVITY DETECTION IN DENTAL IMAGES USING DEEP LEARNING

WONG GUAN TING

This project is submitted in partial fulfilment of the requirements for the degree of
Bachelor of Computer Science with Honours (Computational Science)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2025

**PENGESANAN KAVITI DALAM IMEJ PERGIGIAN
MENGUNAKAN PEMBELAJARAN DALAM**

WONG GUAN TING

Project ini merupakan salah satu keperluan untuk Ijazah
Sarjana Muda Sains Komputer dan Teknologi Maklumat

Fakulti Sains Komputer dan Teknologi Maklumat
UNIVERSITI MALAYSIA SARAWAK
2025

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the Faculty of Computer Science and Information Technology for granting me the opportunity to undertake and complete my final year project. I am truly thankful for the support and resources provided throughout this journey, which have been instrumental in shaping my learning experience.

My deepest appreciation goes to my supervisor, Dr. Chiew Kang Leng, for his unwavering support, insightful guidance, and constant encouragement throughout the course of the project. His expertise and mentorship were invaluable and played a vital role in the successful completion of this work.

I would also like to sincerely thank my examiner, Dr. Hamimah binti Ujir, for her constructive feedback and valuable suggestions. Her input greatly contributed to the improvement and refinement of the system developed in this project.

Lastly, I extend my sincere thanks to my family and friends for their steadfast support, patience, and motivation. Their belief in me provided the strength and inspiration needed to see this project through to completion.

Table of Contents

ACKNOWLEDGEMENT	4
LIST OF FIGURES.....	7
LIST OF TABLES	9
ABSTRACT.....	10
1. CHAPTER 1 INTRODUCTION.....	11
1.1 Background	11
1.2 Problem Statement.....	12
1.3 Scope.....	13
1.4 Objectives.....	13
1.5 Brief Methodology	13
1.6 Significant of the Project.....	15
1.7 Project Schedule	15
1.8 Project Outcome	16
1.9 Project Outline.....	16
2. CHAPTER 2 LITERATURE REVIEW	17
2.1 Introduction.....	17
2.2 Comparison of Deep Learning Models	18
2.2.1 Convolutional neural network.....	19
2.2.2 Faster Region-Convolutional Neural Network.....	20
2.2.3 Xception.....	21
2.2.4 DenseNet121	22
2.2.5 You Only Look Once.....	23
2.2.6 Visual Geometry Group	24
2.2.7 Attention Mechanisms	25
2.3 Review of Related Works	26
2.3.1 Convolutional neural network.....	26
2.3.2 Faster Region-Convolutional Neural Network.....	27
2.3.3 You Only Look Once.....	28
2.3.4 Fourier Descriptors and Bayesian Classification.....	29
2.3.5 Summary of Related Works	30
2.4 Evaluation Metrics.....	31
2.5 Conclusion	32
3 CHAPTER 3 METHODOLOGY	34
3.1 Introduction	34
3.2 Data Collection.....	34
3.3 Data Preprocessing	35
3.4 Model Training and Enhancement.....	37

3.5	Model Evaluation	39
3.6	Conclusion	40
4	CHAPTER 4 IMPLEMENTATION AND TESTING	41
4.1	Introduction	41
4.2	Data Preprocessing	41
4.2.1	Data Annotation Using Roboflow	41
4.2.2	Data Augmentation Using Roboflow	42
4.2.3	Model Training.....	43
4.3	Optimization.....	46
4.4	Results and Analysis.....	48
4.4.1	Comparison of Fine-Tuning Results.....	48
4.4.2	Comparison between Model YOLOv8n and GW-YOLOv8n.....	51
4.5	System Development	52
4.5.1	Home Page	52
4.5.2	Image Detection	53
4.5.3	Video Detection.....	55
4.5.4	Real Time Detection.....	58
5	CHAPTER 5 CONCLUSION AND FUTURE WORK	59
5.1	Introduction	59
5.2	Achievements	59
5.3	Limitations	60
5.4	Future Works.....	61
	References.....	64

LIST OF FIGURES

Figure 1.1: The process flow of models	14
Figure 1.2: The gantt chart of the project	15
Figure 2.1 : Relationship between AI, ML and DL	17
Figure 2.2: Several CNN layers	19
Figure 2.3: Comparison of network structures between Fast RCNN and Faster RCNN (Oppenheim et al., 2018)	20
Figure 2.4: Depthwise convolution and pointwise convolution (Di et al., 2023).....	21
Figure 2.5: The basic block diagram for the Xception block architecture (Guo et al., 2020).....	21
Figure 2.6: Structure of Dense Block (Song et al., 2024).....	22
Figure 2.7: Summary of different DenseNet network structures (Albahli et al., 2021).....	22
Figure 2.8: Description of bounding box.	23
Figure 2.9: The graphical representation of YOLO's workflow (Lysechko et al., 2024).....	24
Figure 2.10: Architecture for VGG16 & VGG19 (Simonyan & Zisserman, 2014)	24
Figure 2.11: Comparison between soft attention and hard attention (Lanham, 2021).....	25
Figure 2.12: Snapshot of the dataset used (Sonavane et al., 2021)	26
Figure 2.13: Testing result for the model (Laishram & Thongam, 2020).....	27
Figure 2.14: The bounding box in different colors was annotated (Marzieh Razaghi et al., 2024)	28
Figure 2.15: Dental classification method diagram (Mahoor & Abdel-Mottaleb, 2005)	29
Figure 3.1: Flow chart of methodology	34
Figure 3.2: Example dataset from Kaggle	35
Figure 3.3: The structure diagram of YOLOv8.....	37
Figure 3.4: The overview of Global Attention Mechanism (Yi et al., 2024).....	37
Figure 3.5: The structure diagram of GW-YOLOv8n	38
Figure 3.6: The schematic diagram of CIoU and WIoU (Shi et al., 2024)	39
Figure 4.1: The dataset was uploaded to Roboflow	41
Figure 4.2: Manually drawing bounding boxes for the intraoral images	42
Figure 4.3: The dataset was split into three subset	43
Figure 4.4: Export augmented dataset in YOLOv8 format.....	43
Figure 4.5: Modified YOLOv8n backbone with the integration of the GAM Attention module	44
Figure 4.6: Adding new class for WIoU in metric python file.....	45
Figure 4.7: Replacement of the iou and loss_iou functions	45
Figure 4.8: Initial model training for proposed GW-YOLOv8n	46
Figure 4.9: Best optimization results of the GW-YOLOv8n	48
Figure 4.10: Validate the optimized model using validation dataset.....	49

Figure 4.11: Testing the optimized model using testing dataset	49
Figure 4.12: The detection outcome of unoptimized model and optimized model	50
Figure 4.13: The detection outcome of proposed model and YOLOv8n.....	51
Figure 4.14: Homepage for the dental cavity detection system	53
Figure 4.15: Implementation of homepage using React.js.....	53
Figure 4.16: Cavity image detection results using the YOLOv8 model	54
Figure 4.17: Front end sent POST request to process image detection	54
Figure 4.18: The backend FastAPI implementation for handling image detection	55
Figure 4.19: Function used to predict uploaded image using proposed model	55
Figure 4.20: Processed video output with detected cavities highlighted	56
Figure 4.21: Code implementation sent POST request to process uploaded video	56
Figure 4.22: Main function to process detection on uploaded video.....	57
Figure 4.23: Progress bar was displayed during processing video detection	57
Figure 4.24: POST request every second to update progress of video detection	58
Figure 4.25: Real time detection for the proposed model.....	58

LIST OF TABLES

Table 2.1: Detailed summary of the deep learning techniques in various study	30
Table 4.1: The experimental environment for the model	44
Table 4.2: The key parameters we used during initial model training.....	46
Table 4.3: The modified key parameters during model optimization.....	47
Table 4.4: The key parameters used for best model optimization.....	47
Table 4.5: Comparative analysis between optimized and unoptimized models in validation datasets	49
Table 4.6: Comparative analysis between optimized and unoptimized models in testing datasets.....	49
Table 4.7: Comparative analysis between GW-YOLOv8n and YOLOv8n in validation dataset	51
Table 4.8: Comparative analysis between GW-YOLOv8n and YOLOv8n in testing dataset	51
Table 5.1: Objectives and achievements of the proposed dental caries detection system	59

ABSTRACT

Dental cavities is a prevalent oral health issue, particularly in underserved and rural communities where access to professional dental care is limited. This study proposes a deep learning-based approach for automated cavity detection in intraoral dental images. A modified YOLOv8 object detection model was trained using a dataset comprising various dental conditions, sourced from public repositories and manually curated images. The methodology involved data preprocessing techniques such as resizing, normalization, and augmentation to enhance model generalization. The system was designed as a web-based platform, enabling users to upload images for cavity detection. Model performance was evaluated using accuracy, precision, recall, and F1-score, demonstrating significant potential for clinical and non-clinical applications. Results indicate that the proposed system effectively identifies cavities with accuracy of 89%, offering a cost-effective, accessible solution for early caries detection. This research contributes to the advancement of AI-driven dental diagnostics, reducing the dependency on expensive equipment and improving oral healthcare accessibility.

Keywords: Cavity Detection, Deep Learning, YOLOv8, Dental Caries, Image Processing, Computer Vision

CHAPTER 1 INTRODUCTION

1.1 Background

Dental cavities, commonly referred to as dental caries, are a prevalent oral health issue that impacts individuals of all ages. Untreated dental caries in permanent teeth were reported as the most widespread health condition in the Global Burden of Disease study 2019, impacting approximately 2 billion people and showing an over 45% rise in cases from 1990 to 2019 (Wen et al., 2021). However, the annual report on dental attendance by state revealed that only 11.5% of adults in Pahang attended primary oral healthcare (Mohd et al., 2021). According to the research conducted by Jasmin & Jaafar. (2011), among the Malaysian Territorial Army, it indicates that only 13.7% of individuals attended regular dental check-ups, despite the services being offered free of charge. The prevalence may be considerably lower in low- and middle-income countries due to the high costs associated with dental examinations and the absence of insurance coverage (Bernabé et al., 2017).

Dental cavities are typically diagnosed using various medical tools and diagnostic techniques, each with its own level of accessibility and accuracy. Traditional methods like visual inspection, tactile probing, and X-ray imaging are widely used in dental practice. Visual inspection and probing are cost-effective but can be inaccurate for early-stage caries, especially in hidden areas (Angelone et al., 2023). According to Albano et al. (2024), X-rays are useful for detecting deeper decay, involve radiation exposure and require specialized equipment, which can be a barrier in underserved regions. Advancements in technology have led to the development of more precise and non-invasive tools such as intraoral scanners and laser fluorescence devices. Intraoral scanners are increasingly utilized not only to capture 3D digital representations of the oral cavity but also to detect dental caries through metrological analysis and their integration with radiographic techniques (Michou et al., 2021). However, their diagnostic capabilities are still being evaluated, and more clinical evidence is needed to validate their effectiveness over traditional methods.

Artificial intelligence (AI) has also been integrated into dental diagnostics, particularly for analyzing radiographic images for caries detection. AI models, especially those based on convolutional neural networks (CNNs), have shown promising accuracy in

identifying carious lesions. These models can process large datasets of dental images, offering a potential solution for automated detection with sensitivity and specificity comparable to experienced dental professionals.

1.2 Problem Statement

Despite significant advancements in dental healthcare, the World Health Organization (WHO) reports that rural areas frequently lack primary dental care services, leading to substantial health disparities and increased rates of untreated caries (World Health Organization, 2023). The economic burden of diagnostic dental equipment, such as digital radiography and Cone-Beam Computed Tomography (CBCT), further exacerbates this issue. According to Vujicic et al. (2016), the financial strain associated with these tools limits their availability primarily to urban centers, where healthcare resources are concentrated. As a result, rural populations rely on outdated diagnostic techniques or receive no diagnosis, leading to an increased prevalence of untreated caries and associated complications (Centers for Disease Control and Prevention, 2024).

To mitigate this gap, a cost-effective and accessible system for early detection of dental caries is essential. However, the development of such a system poses technical challenges. Currently caries detection technologies, including those based on deep learning, require large annotated datasets, high-quality images, and significant computational power. Tooth decay can sometimes be misinterpreted as shadows due to various factors, including low image quality. Additionally, many Orthopantomogram (OPG) images suffer from inadequate clarity and do not distinctly display anatomical structures, which can be attributed to low contrast, uneven exposure, and other image quality issues (Ayat AlSayed et al., 2023). Given these challenges, there is a clear need for a robust, user-friendly, and technically adaptable system that empowers non-specialist users in rural areas to detect dental caries early. Addressing these technical and social needs can help bridge the dental healthcare gap and improve oral health outcomes in underserved communities.

1.3 Scope

This project will employ a modified YOLOv8 model specifically designed to detect and classify dental cavities in non-X-ray images that show the entire mouth rather than isolated images of single teeth. This broader image scope allows for comprehensive analysis, enabling the model to identify carious lesions across multiple teeth in a single, non-specialized image. The model's performance will be evaluated using key metrics such as accuracy, recall, precision, and F1 score to ensure reliable detection across varying image conditions. An interface will be developed to make the system accessible for non-specialist users, such as health workers in remote areas, facilitating early and efficient detection of dental cavities for underserved populations.

1.4 Objectives

- (1) To investigate computer vision and deep learning approaches for cavity detection.
- (2) To develop a system using image processing and deep learning techniques for detecting dental caries
- (3) To evaluate the performance of the proposed system in terms of detection accuracy.

1.5 Brief Methodology

The methodologies for dental cavity detection involve a series of techniques and processes tailored to accurately identify cavities in intraoral images. These methodologies consist of several steps, including data collection and preprocessing, model selection, training, and evaluation.

Firstly, intraoral dental images are gathered from a combination of public repositories and manually curated datasets representing various cavity conditions. The collected data undergo preprocessing steps such as resizing to standardize dimensions, normalization to align pixel intensity levels, and data augmentation techniques like rotation, flipping, and brightness adjustments to improve the model's robustness and generalization.

Besides, the suitable machine learning model is selected based on the prediction goal. In this case, a modified YOLOv8 object detection model is chosen for its ability to efficiently classify and localize dental cavities in intraoral images. This model is particularly suitable due to its real-time detection capabilities and adaptability to challenges such as poor contrast and uneven exposure in dental imagery. After that, the preprocessed intraoral image dataset is used to train it, enabling the model to learn and identify patterns and relationships within the dental images, such as variations in texture, shape, and intensity indicative of cavities.

Furthermore, for the task of dental cavity detection, various evaluation metrics are used to assess the model's performance, including accuracy, precision, recall, F1-score, and mean squared error. Cross-validation techniques are applied to evaluate the model on multiple subsets of the dataset, providing a comprehensive understanding of its prediction capabilities and reliability. A simple process flow is shown in Figure 1.1.

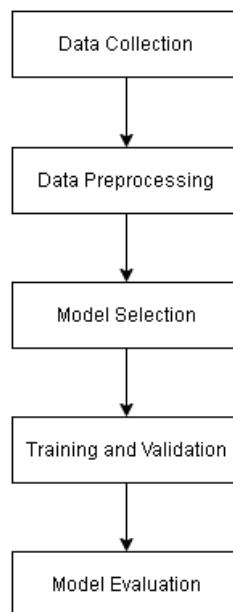


Figure 1.1: The process flow of models

1.6 Significant of the Project

By integration deep learning and image processing techniques, this project aims to revolutionize the way tooth caries is detected by people in underserved and rural areas. The project addresses the pressing need for affordable dental diagnostics, particularly in regions where dental care services are scarce or too costly, thus providing users with a powerful tool and making early caries detection more accessible. This initiative has the potential to significantly enhance public health by making dental care more inclusive and widely available.

1.7 Project Schedule

The project schedule consists of Final Year Project 1 and Final Year Project 2. As in Figure 1.2, the project schedule consists of each task to be done throughout the Final Year Project and visual in gantt chart.



Figure 1.2: The gantt chart of the project

1.8 Project Outcome

A web-based system will be developed to enable users, including non-specialists, to detect dental cavities with minimal difficulty. The system will allow users to upload a mouth image, upon which the modified YOLOv8 model will automatically analyze and classify each visible tooth as either healthy or affected by caries. Detected cavities will be highlighted in the output, providing a clear indication of each tooth's health status. Additionally, the system will display a probability score for each tooth's classification, indicating the confidence level of the model's prediction. This feature enables users to better understand the likelihood of each diagnosis, supporting more informed decision-making.

1.9 Project Outline

Chapter 1 introduces the project by providing an overview of its background, defining the problem statement, and outlining the scope and objectives. It also describes the methodology in brief, highlights the project's significance, presents a schedule, and outlines the anticipated outcomes.

Next, Chapter 2 delves into the literature review, offering insights into the fundamentals of prediction in machine learning, a comparative analysis of various machine learning models, the tools available for implementation, and related projects in the field.

Furthermore, Chapter 3 explains the methodology in detail, including steps such as data collection, preprocessing, model selection, testing, and evaluation.

In addition, Chapter 4 focuses on results and discussion, interpreting the outputs of the models, comparing their performances, and identifying both their strengths and limitations.

Lastly, Chapter 5 concludes the project with a summary of the findings, discusses the study's implications, and provides recommendations for future research.

CHAPTER 2 LITERATURE REVIEW

In Chapter 2, we will explore similar projects that have utilized deep learning for dental cavity detection. This review will provide insights into the methodologies they employed, highlighting their successes and limitations. We will compare different machine learning models to identify approaches that may be most effective for our task of cavity detection. Additionally, we will examine the tools and software used to develop and implement these models, such as frameworks for object detection and deep learning. This chapter aims to leverage existing research and practices to guide the development of our project, ensuring informed and optimized decisions.

2.1 Introduction

Machine Learning (ML) is a prominent branch of artificial intelligence (AI) that focuses on enabling computers to learn from data and make predictions or decisions without explicit programming (Piuri et al., 2020). Machine learning involves the development of algorithms and models to identify data patterns and insights to make predictions, decisions and automate tasks.

Machine learning is integral to various industries due to its numerous applications. It leverages historical data for predictions and forecasting, automates repetitive tasks, identifies patterns and trends within large datasets, personalizes user experiences through recommendation systems, and optimizes processes by detecting inefficiencies and suggesting enhancements.

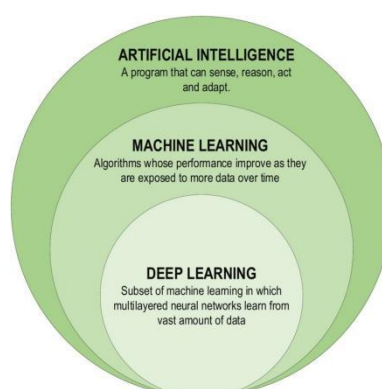


Figure 2.1 : Relationship between AI, ML and DL

Deep learning (DL) is a specialized subset of machine learning that uses artificial neural networks to simulate the functioning of the human brain (Sarker, 2021). Deep learning employs multiple layers of interconnected nodes, or neurons to learn data representations at various levels of abstraction. In contrast to traditional machine learning approaches, Deep learning models autonomously extract pertinent features directly from raw data (Taye, 2023). This capability renders them particularly effective for handling unstructured inputs, including images, text, and audio without the need for manual feature engineering. Deep learning drives a wide range of advanced applications, such as image recognition, language translation, speech processing and autonomous systems which highlights its pivotal role in advancing technology across multiple fields (Mehrish et al., 2023).

The deep learning prediction or classification process begins with the collection and preparation of a substantial dataset. This involves cleaning the data to remove inconsistencies and transforming it into a suitable format for deep learning frameworks. Relevant features are not manually selected, as deep learning models inherently extract hierarchical features directly from the raw data. The machine learning models is chosen, and it is trained to identify patterns on the prepared dataset. During training, the model's performance is validated on a separate dataset to ensure generalization and avoid overfitting.

Assessing the model's performance, incorporating new data for retraining, and fine-tuning hyperparameters constitute integral components of the continuous improvement process. Deep learning leverages learned representations from the training data to make accurate predictions or classifications on new instances. The accuracy and reliability of predictions are influenced by the volume and diversity of data, the complexity of the model architecture, and the implementation of effective training methodologies (Hu et al., 2021). The model's deployment enables object detection, language translation, improving decision-making and automation (Manakitsa et al., 2024).

2.2 Comparison of Deep Learning Models

Convolutional Neural Network (CNN), Faster Region-Convolutional Neural Network (Faster R-CNN), You Only Look Once (YOLO), Xception, DenseNet121 and Visual

Geometry Group (VGG) are some commonly used deep learning models. These models will be further discussed in the following sections.

2.2.1 Convolutional neural network

Convolutional Neural Network (CNN) is advanced AI models composed of multiple layers, designed to detect, identify, classify, and segment objects within images with high accuracy (Taye, 2023). CNN is a widely used discriminative deep learning architecture that can learn directly from input data without requiring human feature extraction (Alzubaidi et al., 2021). The architecture of CNN draws inspiration from the neural structures observed in the human and animal brain, mirroring the foundational principles of conventional neural networks.

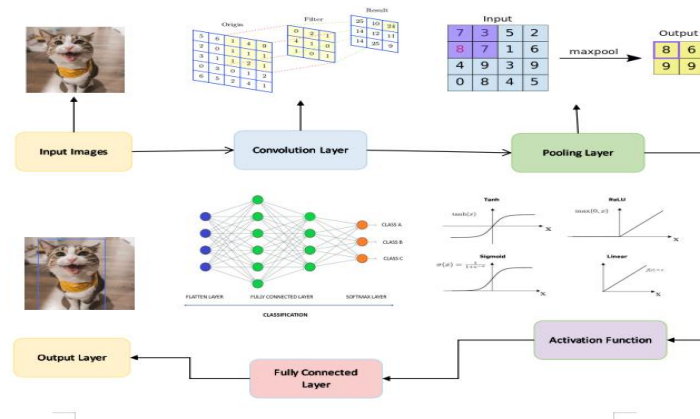


Figure 2.2: Several CNN layers

The convolutional layer extracts local features such as edges and textures by applying learnable filters through a process called convolution (Moez Krichen, 2023). The pooling layer reduces the spatial dimensions of feature maps, lowering computational complexity and preventing overfitting while retaining essential information (Zafar et al., 2022). The activation function layer introduces non-linearity, allowing the network to capture complex patterns that a linear model cannot (Jagtap & Karniadakis, 2023). Finally, the fully connected layer synthesizes high-level features extracted by previous layers to make final predictions for tasks such as classification or regression (Yamashita et al., 2018).

2.2.2 Faster Region-Convolutional Neural Network

Faster Region-Based Convolutional Neural Network (Faster R-CNN) is an advanced object detection framework that integrates region proposal generation and object classification into an end-to-end trainable model (Ren et al., 2017). Faster R-CNN builds upon the foundation of Fast R-CNN by introducing improvements that address the slow candidate box selection issue inherent in the Selective Search algorithm. Instead of relying on an external method for region proposal creation as in Fast R-CNN, Faster R-CNN incorporates a Region Proposal Network (RPN) which processes the feature map from the convolutional layers and generates region proposals directly (Figure 2.3).

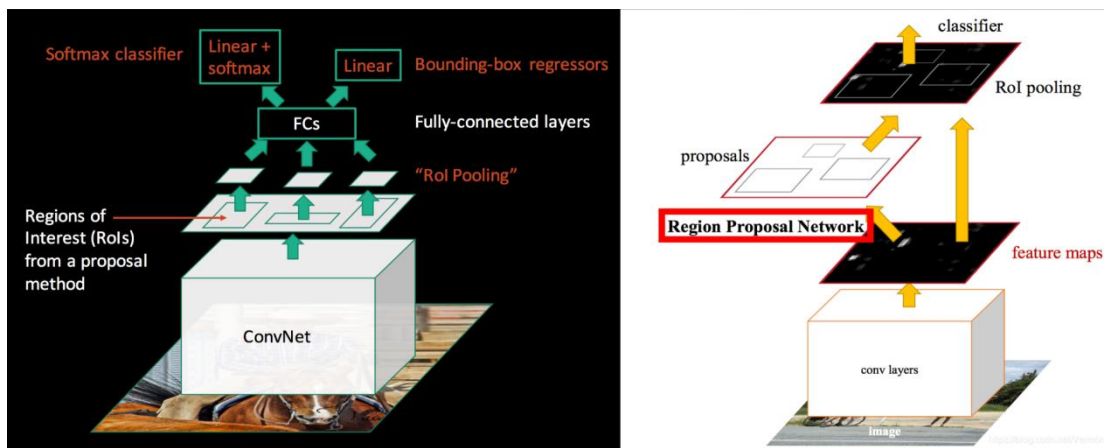


Figure 2.3: Comparison of network structures between Fast RCNN and Faster RCNN (Oppenheim et al., 2018)

The process begins with convolutional, ReLU, and pooling layers extracting spatial and semantic features from input images to create feature maps. These maps are shared with the Region Proposal Network (RPN), which uses sliding windows and softmax functions to generate and refine region proposals via bounding box regression. The RoI pooling layer aligns and resizes these proposals with the feature maps, preparing them for further processing. Finally, fully connected layers predict object classes and refine bounding box coordinates, ensuring accurate object localization and classification.

2.2.3 Xception

Xception short for “Extreme Inception”, is an improved model proposed by François Chollet following Inception, specifically as a refinement of InceptionV3. Xception replaces traditional Inception modules with depthwise separable convolutions, effectively factorizing standard convolution operations into two steps which are depthwise convolution and pointwise convolution (Chollet, 2016).

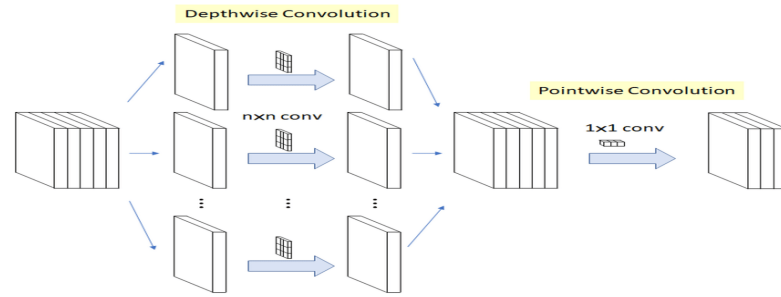


Figure 2.4: Depthwise convolution and pointwise convolution (Di et al., 2023)

One filter is applied per input channel rather than applying a single convolutional kernel across all input channels in depthwise convolution. This allows each channel to capture spatial features independently. For an input with C channels, this results in C separate convolutions, each using its own $N \times N$ kernel. After the depthwise convolution, a 1×1 pointwise convolution is applied across all the channels. This step combines the spatial features captured by the depthwise convolution and learns the relationships between the different channels. The pointwise convolution produces the desired number of output feature maps, effectively merging the information from each channel while maintaining efficiency.

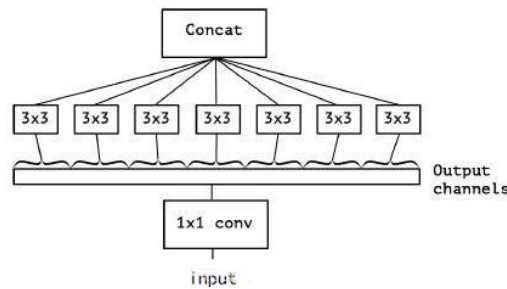


Figure 2.5: The basic block diagram for the Xception block architecture (Guo et al., 2020)

The Xception model modified the original Inception block by expanding its width and replacing a single 3×3 convolution with a depthwise separable convolution followed by a 1

$\times 1$ convolution to reduce computational complexity (Zhou et al., 2022). Xception avoids using ReLU activation layers between the operations, whereas MobileNet incorporates ReLU layers to preserve the integrity of the data during this process.

2.2.4 DenseNet121

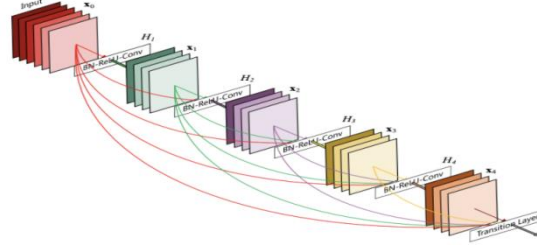


Figure 2.6: Structure of Dense Block (Song et al., 2024)

DenseNet121, part of the DenseNet family which is a deep convolutional neural network introduced to mitigate the vanishing gradient problem and significantly reduces the number of parameters compared to traditional convolutional networks (Huang et al., 2017). DenseNet121 introduces dense connectivity, where each layer is directly connected to every other layer in the same dense block, leading to better network training and reduced overfitting.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 $\times$ 112	7 $\times$ 7 conv, stride 2			
Pooling	56 $\times$ 56	3 $\times$ 3 max pool, stride 2			
Dense Block (1)	56 $\times$ 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 $\times$ 56	1 $\times$ 1 conv			
	28 $\times$ 28	2 $\times$ 2 average pool, stride 2			
Dense Block (2)	28 $\times$ 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 $\times$ 28	1 $\times$ 1 conv			
	14 $\times$ 14	2 $\times$ 2 average pool, stride 2			
Dense Block (3)	14 $\times$ 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 $\times$ 14	1 $\times$ 1 conv			
	7 $\times$ 7	2 $\times$ 2 average pool, stride 2			
Dense Block (4)	7 $\times$ 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 $\times$ 1	7 $\times$ 7 global average pool			
		1000D fully-connected, softmax			

Figure 2.7: Summary of different DenseNet network structures (Albahli et al., 2021)

The model starts with an initial convolution and pooling layer, followed by four dense blocks separated by transition layers. Each dense block contains multiple densely connected layers, each consisting of batch normalization, ReLU activation, and convolutions. The transition layers, which include 1x1 convolutions and average pooling, reduce the feature map size and manage the model's complexity. The final global average pooling layer connects to a fully connected layer for classification. DenseNet121's parameters are

configured to achieve a depth of 121 layers, calculated from its specific design of dense and transition layers. The architecture's efficient parameter usage and reduced computation requirements make it suitable for tasks on resource-constrained devices.

2.2.5 You Only Look Once

Real-time object detection has become an essential aspect of diverse applications, encompassing fields such as autonomous driving, robotics, and video monitoring systems (Terven et al., 2023). The You Only Look Once (YOLO) algorithm, proposed by Joseph Redmon and colleagues in 2016, stands for "You Only Look Once" which represents a significant advancement in object detection and renowned for its exceptional balance between speed and accuracy (Redmon et al., 2016). In contrast to conventional two-stage detectors that distinctively separate region proposal and classification tasks, YOLO employs an integrated approach which executing both tasks simultaneously within a single pass through the network (Hussain & Khanam, 2024).

According to Prinzi et al. (2023), the core concept of YOLO involves partitioning the input image into a grid of size $S \times S$, where each grid cell predicts bounding box coordinates, objectness scores, and class probabilities. The bounding boxes are characterized by four descriptors which are center of a bounding box (b_x, b_y), width (b_w), height (b_h), value c which corresponds to the class of an object and a confidence score (Vig et al., 2022). The confidence score represents the likelihood that a bounding box contains an object and the accuracy of the predicted coordinates, computed as the product of the probability of the object and the Intersection over Union (IOU) between the predicted and ground truth bounding boxes (Terven et al., 2023). In addition, it is necessary to predict the probability that an object is present within the bounding box, represented as $y = (p_c, b_x, b_y, b_w, b_h, c)$. The final detections are obtained by combining the class probabilities with the bounding box confidence scores, assigning the most probable class to each box.

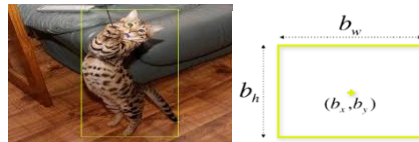


Figure 2.8: Description of bounding box.

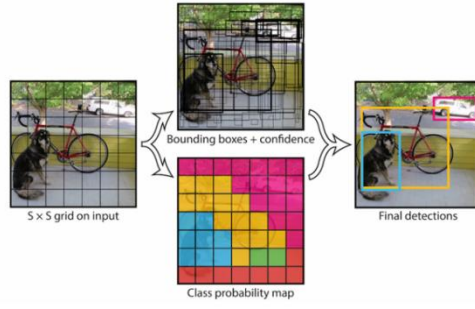


Figure 2.9: The graphical representation of YOLO's workflow (Lysechko et al., 2024)

2.2.6 Visual Geometry Group

VGG16 and VGG19 are deep convolutional neural networks proposed by the Visual Geometry Group (VGG) at the University of Oxford. These architectures were introduced in the seminal paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" and achieved state-of-the-art performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 (Simonyan & Zisserman, 2014).

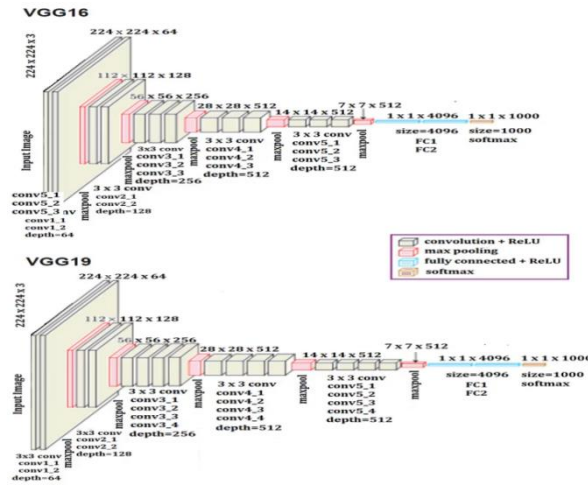


Figure 2.10: Architecture for VGG16 & VGG19 (Simonyan & Zisserman, 2014)

VGG16 comprises 16 layers of learnable parameters, including 13 convolutional layers and 3 fully connected layers, while VGG19 extends this architecture by adding three more convolutional layers to 19 layers. Both models use max pooling layers for spatial down-sampling and Rectified Linear Unit (ReLU) activations for non-linearity, making them computationally efficient despite their depth. VGG networks paved the way for deeper architectures by demonstrating that smaller convolutional filters could achieve superior performance compared to larger filters used in earlier models. However, their large model size and high computational cost limit their application in resource-constrained environments

(Maji & Mullins, 2018). Despite this, VGG16 and VGG19 remain influential benchmarks and are widely used as feature extractors in transfer learning tasks due to their robust pre-trained weights on ImageNet.

2.2.7 Attention Mechanisms

The attention mechanism has emerged as a transformative approach in deep learning, enhancing the capabilities of models to selectively focus on relevant parts of the input data. Initially introduced in natural language processing tasks, attention mechanisms have found extensive applications in computer vision, enabling models to achieve higher accuracy and efficiency in complex tasks, including object detection, image segmentation, and classification.



Figure 2.11: Comparison between soft attention and hard attention (Lanham, 2021)

Attention mechanisms enable neural networks to prioritize critical features from the input data by assigning weights to different regions or elements (Duhan et al., 2024). This selective focus ensures that the model dedicates computational resources to the most informative aspects of the input, reducing noise and improving interpretability. Broadly, attention mechanisms can be categorized into soft attention and hard attention. Soft attention assigns varying degrees of importance to all input elements, maintaining differentiability, while hard attention selects specific elements, often requiring reinforcement learning for optimization.

2.3 Review of Related Works

The use of deep learning in healthcare has proven instrumental in addressing various challenges by enabling accurate and cost-effective diagnostic solutions. To enhance understanding of dental cavity detection, this section will review related works by other researchers that focus on the application of machine learning for dental health diagnostics and object detection in medical imaging. These studies provide valuable insights into the methodologies, models, and tools employed in similar projects, serving as a foundation for developing an optimized approach for this project.

2.3.1 Convolutional neural network

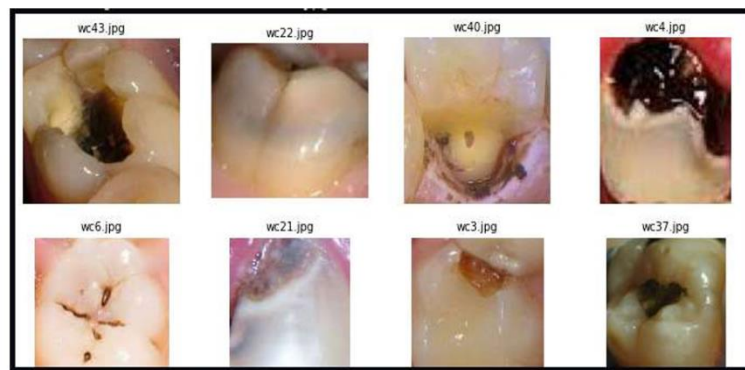


Figure 2.12: Snapshot of the dataset used (Sonavane et al., 2021)

The study by Sonavane et al. (2021) explored the classification of dental cavities using CNNs and contributed to the understanding of machine learning applications in oral healthcare. The study relied on a dataset sourced from Kaggle, consisting of images categorized as caries or non-caries teeth. The data was split into training and testing sets in an 80:20 ratio, with the training set used to optimize a CNN model. While the model achieved a maximum accuracy of 71.43%, the authors suggested that increasing the size of dataset and further refining the model could significantly enhance its diagnostic accuracy.

2.3.2 Faster Region-Convolutional Neural Network

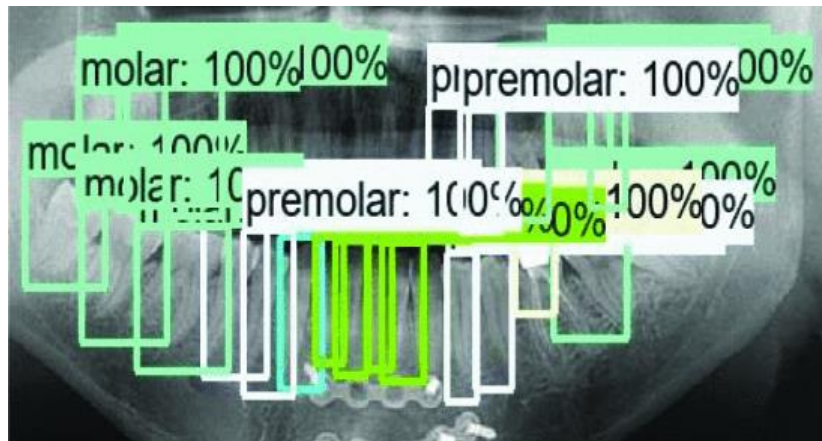


Figure 2.13: Testing result for the model (Laishram & Thongam, 2020)

The study by Laishram and Thongam (2020) focused on detecting and classifying dental pathologies using Faster R-CNN on Orthopantomogram (OPD) radiographs. The research aimed to classify different types of teeth, such as incisors, molars, premolars, and canines, as well as detect dental anomalies like fixed partial dentures and impacted teeth. The dataset included 96 training images and 49 testing images, with manual annotation used to generate labels. Their model achieved a detection accuracy exceeding 90% and a classification accuracy of 99%, demonstrating the effectiveness of the approach.

The use of Faster R-CNN contributed to the model's speed and precision, achieving high accuracy for both detection and classification tasks. However, the study also had limitations. The dataset was relatively small, comprising only 145 images, which may impact the model's generalizability to broader populations and varied conditions. Furthermore, the images were sourced from a single clinic, introducing potential bias and limiting the applicability of the model to other imaging equipment or patient demographics. Additionally, the reliance on manually labelled data could introduce labor-intensiveness and potential human error. The authors suggested that expanding the dataset and further refining the algorithm could enhance the model's accuracy and applicability.

2.3.3 You Only Look Once

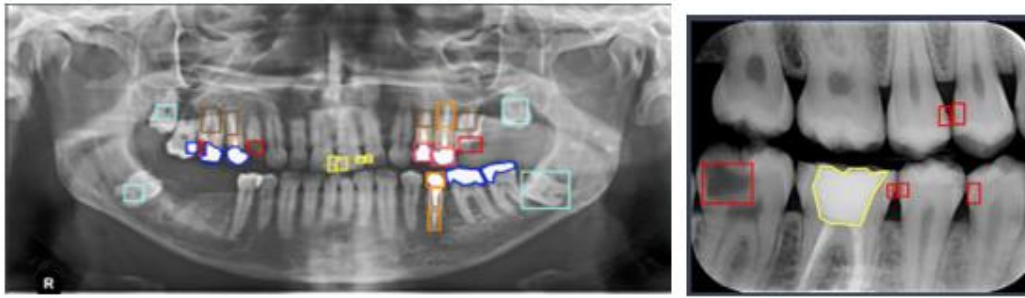


Figure 2.14: The bounding box in different colors was annotated (Marzieh Razaghi et al., 2024)

The study by Razaghi et al. (2024) presents a deep learning-based approach for diagnosing dental diseases using the YOLOv8 model. The researchers developed a comprehensive dataset consisting of 1,516 images, derived from 316 annotated panoramic and bitewing dental X-rays, expanded through data augmentation. The YOLOv8 model variants were trained to classify 12 dental conditions, including dental caries, crown restorations, and periapical lesions (Figure 2.4). Among the YOLOv8 versions tested, YOLOv8m achieved the best performance with a mean average precision (mAP) of 71.6%, precision of 90%, and recall of 90%. The results highlight the model's potential for accurate and efficient dental diagnostics.

The study's strengths include leveraging YOLOv8's efficient object detection architecture and employing data augmentation to enhance model generalization. However, limitations arise from dataset imbalance and challenges in recognizing visually similar dental anomalies. The authors suggest future improvements through better preprocessing algorithms, increased dataset size, and balanced class distribution. They conclude that incorporating advanced models like YOLOv8 can significantly improve dental disease diagnosis, paving the way for robust clinical decision support systems.

2.3.4 Fourier Descriptors and Bayesian Classification

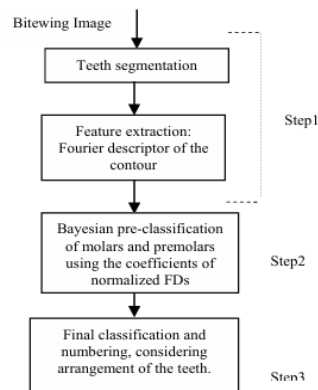


Figure 2.15: Dental classification method diagram (Mahoor & Abdel-Mottaleb, 2005)

The study by Mahoor and Abdel-Mottaleb (2004) introduces a mathematical approach for the automatic classification and numbering of teeth in bitewing dental images. The researchers developed a segmentation algorithm that uses morphological operations and adaptive thresholding to extract tooth contours. Fourier descriptors (FDs) were applied to characterize tooth shapes, offering invariance to translation, scaling, and rotation. Bayesian classification was then used to differentiate molars from premolars, while the spatial arrangement of teeth was utilized to correct misclassifications and assign numbers according to the universal dental numbering system. The method demonstrated high classification accuracy in experiments on 50 bitewing images, though classifying premolars proved more challenging due to overlapping features.

The approach's strengths include the effective use of Fourier descriptors and Bayesian classification, which minimized the need for large labelled datasets, a typical limitation in machine learning. However, the method faced challenges with partial tooth visibility and occasional segmentation inefficiencies, requiring manual intervention in some cases. Despite these limitations, the study emphasized the value of incorporating spatial relationships to enhance classification performance, showcasing the potential for automated dental identification systems. This method offers a promising step toward improving the accuracy and efficiency of dental image analysis for diagnostics and treatment planning.

2.3.5 Summary of Related Works

Table 2.1 offers a detailed summary of the various deep learning techniques applied in various fields, outlining their datasets used, and corresponding performance results.

Table 2.1: Detailed summary of the deep learning techniques in various study

Method Used	Purpose	Dataset	Results	Challenges	Limitations
CNN (Sonavane et al, 2021)	To explore CNNs for classifying dental images into caries and non-caries categories, demonstrating the potential of deep learning in oral health.	Use Kaggle dataset with 60 images for training and 14 for testing.	Achieved 71.43% accuracy.	- Limited dataset size hindered model robustness. - Suggested larger datasets and model refinement for improved performance.	- Relied on a very small dataset, which limited the generalizability of the findings. - Lack of diversity in data impacted model accuracy.
Faster R-CNN (Laishram and Thongam, 2020)	Detect and classify dental pathologies using Orthopantomogram (OPD) radiographs	96 training images 49 testing images (manual annotation)	- Detection accuracy >90% - Classification accuracy 99%	- Small dataset size - Risk of overfitting - Limited generalizability	- Manual labeling could introduce human error - Small dataset (145 images) from a single clinic.
YOLOv8 (Razaghi et al., 2024)	To develop an automated diagnostic system for detecting and classifying dental diseases using panoramic and bitewing X-ray images	A custom dataset of 316 annotated panoramic and bitewing X-ray images expanded to 1,516 samples through data augmentation, including 12 dental classes	Best performance achieved with YOLOv8m: mean average precision (mAP) of 71.6%, precision of 90%, and recall of 90%	Difficulty in recognizing visually similar dental anomalies, variability in X-ray image quality, and noise in panoramic images.	- Imbalanced dataset. - Small dataset size for certain dental classes. - Comprehensive annotations.
Fourier Descriptors and Bayesian Classification (Mahoor and Abdel-Mottaleb, 2004)	To develop an automated system for classification and numbering of teeth in bitewing dental images.	50 bitewing dental images.	High classification accuracy, particularly for molars and premolars with scores above 90%.	- Difficulty in classifying premolars due to overlapping features with other teeth. - Partial tooth visibility in some images, requiring manual intervention.	- Less effective in handling partial tooth visibility. - Occasional segmentation inefficiencies requiring manual correction. - Premolar classification challenging due to overlapping features.

2.4 Evaluation Metrics

The evaluation of the model's performance involves using metrics such as Recall (R), Precision (P), F1 Score, and mean average precision (mAP). These metrics are derived by comparing the model's predictions to the ground truth labels in the dataset. Key components in this assessment include True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). From these values, Precision (P) and Recall (R) can be computed to quantify the model's accuracy and sensitivity.

Precision provides insight into the model's ability to accurately predict positive outcomes. It is calculated as the proportion of positive predictions that are true positives (TP), meaning correctly identified cavity-affected teeth. A high precision score minimizes false positives (FP), indicating that the model is reliable when identifying cavity-affected teeth.

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\%$$

Recall (R) measures the model's ability to accurately identify positive test cases. It assesses the model's effectiveness by indicating the proportion of positive instances correctly detected. A higher sensitivity signifies better performance in correctly predicting positive cases.

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\%$$

The F1 Score (F1) is a performance metric that combines Precision and Recall providing a balanced evaluation of the model's effectiveness. In the context of identifying cavity teeth, a high F1 score reflects the model's capability to accurately detect true cavity while minimizing false positives. Since detecting tooth cavity requires both identifying as many true fissures as possible (high Recall) and ensuring the identified fissures are accurate (high Precision), the F1 score serves as a holistic metric for assessing the model's overall performance. It accounts for both precision and recall and is calculated using the following formula:

$$\text{F1 Score} = 2 \times \frac{R \times P}{R + P}$$

IoU (Intersection over Union) measures the overlap between the predicted bounding box B_p and the ground truth bounding box B_g . It is a crucial metric for evaluating the accuracy of spatial localization of objects.

$$\text{IoU} = \frac{B_p \cap B_g}{B_p \cup B_g}$$

Mean Average Precision (mAP) is a key performance metric used to evaluate the accuracy of predicting target locations and classifications. In our experiments, we utilize mAP as one of the most critical metrics to assess the effectiveness of target detection algorithms. It is determined by calculating the mean of the Average Precision (AP) values. The AP is computed using the following formula:

$$\text{AP} = \int_0^1 P(R) dR$$

For evaluating the object detection models, the mean Average Precision (mAP) was calculated using two Intersection over Union (IoU) thresholds: 0.5 (mAP@0.5) and 0.5:0.95 (mAP@[0.5:0.95]). These thresholds were selected to align with standard practices and ensure comparability with other studies in object detection benchmarks.

2.5 Conclusion

This chapter comprehensively reviewed the application of deep learning in dental cavity detection, comparing methodologies, models, and tools employed in previous studies. Various deep learning models, including CNNs, YOLO, Faster R-CNN, and ensemble approaches, were examined in terms of their implementation, strengths, limitations, and performance metrics. The findings from similar studies highlight potential strategies for optimizing methodology, emphasizing the importance of leveraging efficient models such as YOLO for improved accuracy and time efficiency. To address the specific requirements of dental cavity detection, where accurate identification and real-time inference are essential, the model was optimized for minimal parameters, low memory consumption, and fast processing. Consequently, YOLOv8n was selected as the baseline model for enhancement. This variant features a reduced network depth and width while retaining the core capabilities of the YOLOv8 series, making it well-suited for efficient and precise intraoral image analysis in

clinical and diagnostic applications. This decision reflects YOLOv8n advancements in feature extraction, scalability, and real-time detection capabilities, aligning well with the project's objectives. The insights gained from this review will serve as the foundation for the subsequent chapters, where the methodology and implementation strategy will be developed, building upon identified strengths and addressing existing limitations in the field.

CHAPTER 3 METHODOLOGY

3.1 Introduction

This chapter provides a comprehensive explanation of the project's methodology as shown as Figure 3.1, outlining the processes undertaken from inception to completion. The subsequent section will discuss data collection in detail.

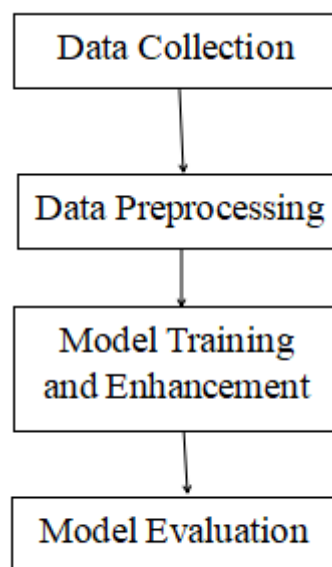


Figure 3.1: Flow chart of methodology

3.2 Data Collection

The dataset used in this study consists of 2244 total intraoral dental images that were sourced from a combination of publicly available repositories and manually curated datasets. This approach ensured a broad representation of dental conditions which includes both cavity-affected and non-cavity-affected teeth, aims to enhance the model's generalization capability across diverse clinical scenarios. Public repositories such as those hosted on platforms like Kaggle (Figure 3.2) served as the primary sources for obtaining these images

(Prajwal Bastewad, 2024). The images were typically downloaded in standard formats, such as JPEG, which facilitated efficient preprocessing and subsequent analysis.

The dataset was specifically designed to exclude certain types of images such as X-rays and segmentation images, as the focus of the study was on detecting cavities in traditional intraoral photographs. This decision aligns with the project's aim to develop a model that can be applied in more accessible, real-world clinical environments where X-ray imaging or detailed segmentation may not be readily available (Ng, 2024). Additionally, images depicting single teeth were not included in the dataset, as the focus was on modeling situations where multiple teeth are visible.

The diverse and comprehensive dataset, comprising a variety of dental conditions which will serve as the foundation for training and evaluating the performance of the modified YOLOv8 model in detecting dental cavities. By incorporating a wide range of image conditions, the model will be better equipped to generalize real-world scenarios, enhancing its accuracy and reliability in detecting cavities under different circumstances.

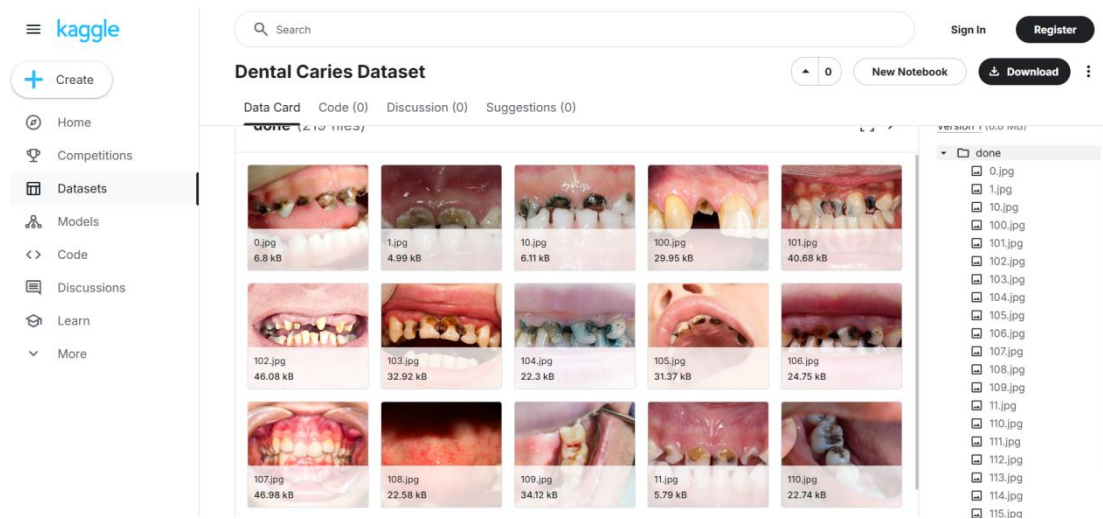


Figure 3.2: Example dataset from Kaggle

3.3 Data Preprocessing

The data preprocessing phase is critical in preparing the collected intraoral dental images for training and evaluating the modified YOLOv8 model. This process ensures that

the input data is of consistent quality, format, and structure, optimizing the model's ability to detect dental cavities accurately and reliably.

Initially, all images were converted into a uniform format and resolution to standardize input dimensions for the model. Resizing was conducted with care to maintain the original aspect ratios, utilizing padding techniques where necessary to avoid distortion of the dental structures. All images, predominantly in JPEG format were resized to a resolution of 640×640 pixels. This resolution was chosen to align with the architectural requirements of the YOLOv8 model, striking a balance between preserving critical image details and computational efficiency.

In addition, data augmentation techniques were applied to increase dataset diversity and reduce the risk of overfitting (Shorten & Khoshgoftaar, 2019). This included random transformations such as rotations, horizontal and vertical flips and adjustments to brightness, exposure and saturation. Additionally, a Gaussian blur was applied to mimic minor camera focus issues or motion blur during image capture. According to BabaAhmadi et al. (2024), Gaussian blur reduces noise in the image, resulting in cleaner and more accurate visual information for improved perception and decision-making. Random noise also was introduced to simulate sensors and environmental noise commonly encountered in real-world photography. These augmentations ensured that the model could generalize better to varying image conditions encountered in real-world environments.

Furthermore, the dataset was split into three subsets: 80% for training, 10% for testing, and 10% for validation. This partitioning ensured that the model was trained on a diverse set of images while reserving a portion for testing its generalization performance and fine-tuning the model's parameters during validation. After augmentation, the dataset was expanded to include 4080 images for training, 487 for testing, and 486 for validation, enhancing the model's robustness and performance.

3.4 Model Training and Enhancement

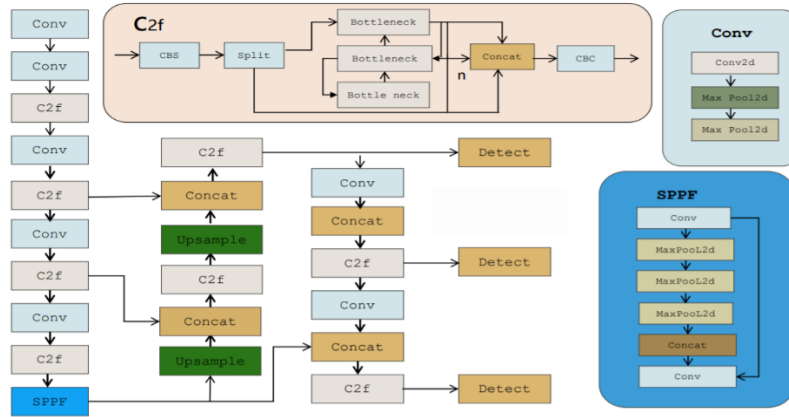


Figure 3.3: The structure diagram of YOLOv8

The YOLOv8 model is a state-of-the-art object detection framework designed for high-speed and accurate detection tasks. YOLOv8 achieves an optimal balance between precision and inference speed, making it well-suited for real-time applications (Hussain, 2024). For the detection of dental caries, the YOLOv8 model was chosen due to its efficiency, scalability, and ability to handle small object detection, which is critical for identifying subtle dental abnormalities. Figure 3.3 shows the structure diagram of the YOLOv8 model.

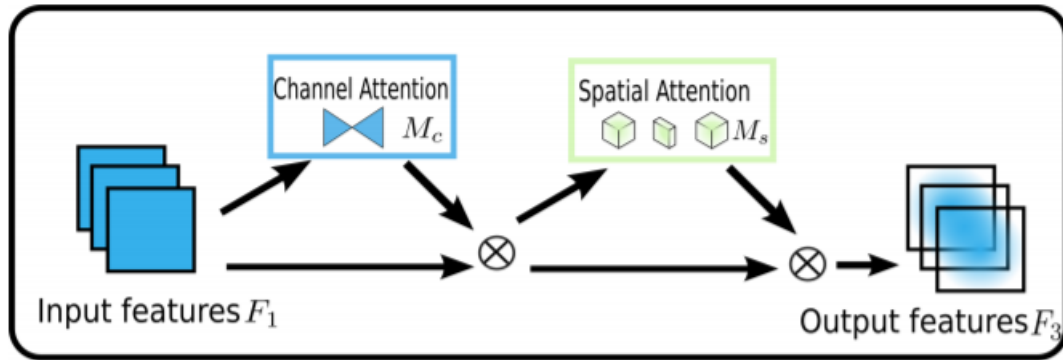


Figure 3.4: The overview of Global Attention Mechanism (Yi et al., 2024)

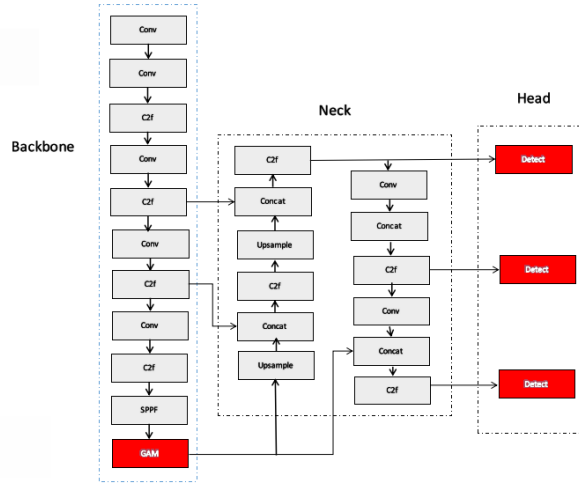


Figure 3.5: The structure diagram of GW-YOLOv8n

Modifications and advanced training strategies were implemented to further enhance the YOLOv8 model for this specialized task. This project proposes a detection mode which is an improvement of YOLOv8 by incorporating Global Attention Mechanism (GAM) and WISE-IoU. The proposed model is termed as GW-YOLOv8n. The primary architectural improvement involved integrating the Global Attention Mechanism (GAM) as shown in Figure 3.4 into the model. GAM allows the model to capture long-range dependencies and focus on the most relevant regions such as areas exhibiting early signs of caries. The modification in Figure 3.5 significantly improves the model's ability to differentiate between caries and other dental features, particularly in complex scenarios where caries may be partially occluded or low contrast.

In addition, various training strategies were employed to optimize the model's performance for tooth caries detection. These included experimenting with learning rate schedules in the range of 0.0001 to 0.1 and fine-tuning key hyperparameters such as batch size. Training experiments were conducted with batch sizes of 10 and 12 to strike a balance between efficient training and avoiding overfitting. Overfitting was monitored throughout training by closely analyzing the training and validation loss curves. The graphs of training loss and validation loss were studied to ensure convergence and identify potential overfitting or underfitting scenarios. When overfitting was observed, adjustments to the batch size, learning rate, and regularization techniques were made to improve generalization. While a larger batch size of 16 was initially considered, it could not be utilized as the laptop's GPU lacked sufficient power to support the higher memory demands.

Additionally, different YOLOv8 pre-trained weight files, including yolov8n.pt and yolov8s.pt were evaluated to identify the most suitable starting point for fine-tuning. These variants differ in model size, computational requirements, and performance, allowing for trade-offs between speed and accuracy.

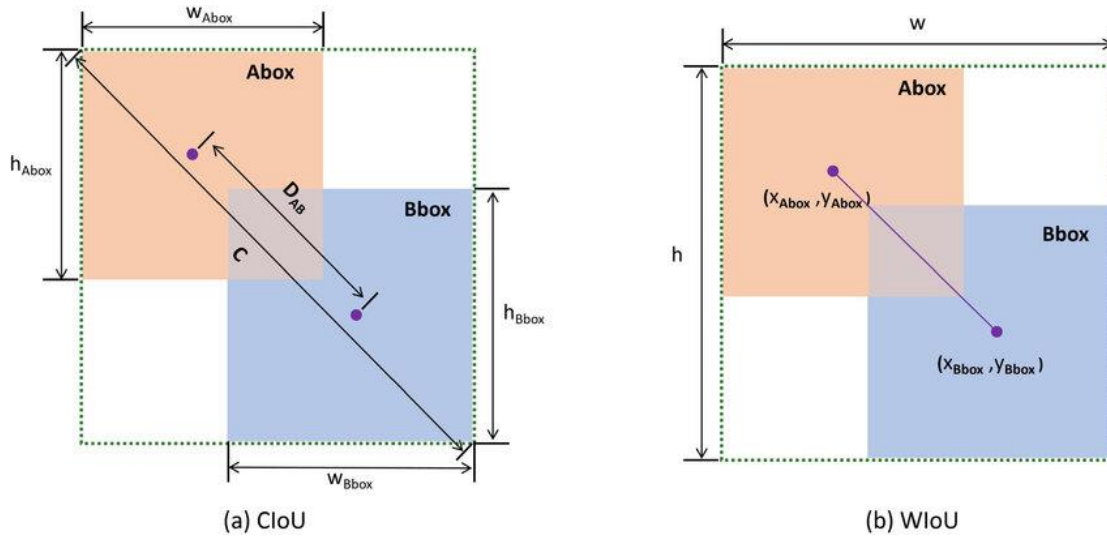


Figure 3.6: The schematic diagram of CIoU and WIoU (Shi et al., 2024)

Alternative loss functions were explored to further enhance the detection accuracy. In addition to the standard IoU loss used in YOLOv8, the Weighted IoU (WIoU) loss function was integrated to improve the model's focus on more critical regions. WIoU assigns higher weights to challenging examples, such as small or partially occluded caries, ensuring the model pays greater attention to these harder-to-detect cases. Other loss functions, such as Complete IoU (CIoU) and Distance IoU (DIoU), were also considered during experimentation to refine the localization of bounding boxes. These enhancements significantly improved the precision of predictions, particularly in detecting early-stage caries.

3.5 Model Evaluation

A structured approach will be adopted to ensure accurate and reliable performance assessment in the model evaluation phase. Firstly, a reserved 10% subset of the dataset will be used exclusively for inference, serving as an unseen test set to validate the model's generalization capabilities. Additionally, K-fold cross-validation will be applied during

training to minimize bias and variance, enabling a thorough evaluation across different subsets of the data.

Fine-tuning of hyperparameters such as confidence threshold, input image size, learning rate, and non-maximum suppression (NMS) values will be carried out iteratively. The model will be retrained and refined until convergence is achieved, balancing accuracy and efficiency. Key evaluation metrics including Precision, Recall, F1 Score, and mean average precision (mAP) as shown in Section 2.4, will be computed to monitor progress. This approach ensures a comprehensive evaluation while addressing potential class imbalance and other dataset-specific challenges.

3.6 Conclusion

In conclusion, the methodology of this project outlines the comprehensive process of detecting dental cavities using deep learning techniques. The data collection phase involved gathering intraoral dental images from reliable sources, representing both cavity-affected and normal teeth. The images were annotated to assign classification labels to each class. The preprocessing stages such as resizing, normalization, and data augmentation were applied to enhance the dataset's quality and diversity. A model was selected and trained for cavity detection. The model's performance was evaluated using metrics such as Precision, Recall, F1 Score, and Mean Average Precision (mAP) which provided a detailed assessment of its ability to accurately identify and localize cavities. The next chapter will focus on the implementation and testing of the model for dental cavity detection.

CHAPTER 4 IMPLEMENTATION AND TESTING

4.1 Introduction

This chapter will delve into the comprehensive process of implementing and testing deep learning models, providing a thorough overview of the project's journey from inception to completion. The subsequent section will explore various critical stages, including data preprocessing, implementation and evaluation of the models. This detailed exploration aims to elucidate each step, ensuring a clear understanding of the methodologies and techniques employed throughout the project.

4.2 Data Preprocessing

The process of annotating a dataset is crucial for training an accurate model for detection. In this section, we will discuss how Roboflow was utilized to preprocess our dataset effectively.

4.2.1 Data Annotation Using Roboflow

Roboflow provides an efficient and user-friendly platform that facilitates dataset annotation, augmentation, and export in YOLOv8-compatible formats. The dataset is uploaded to Roboflow before beginning the annotation process shown as Figure 4.1.

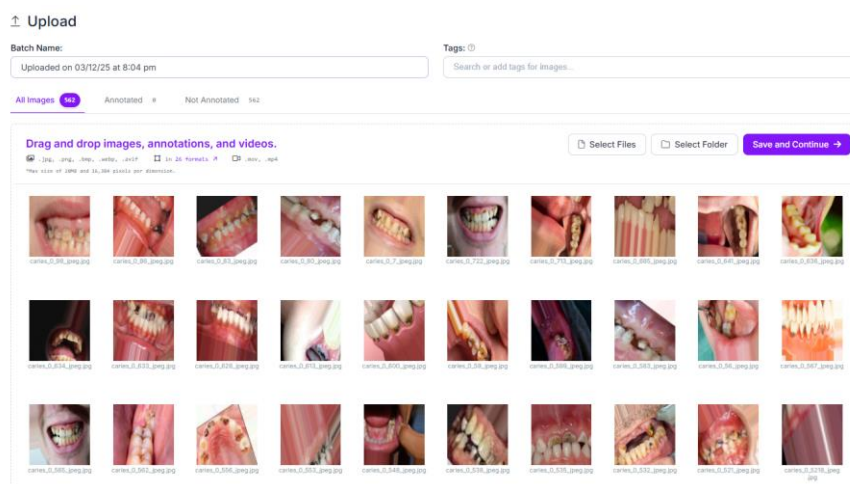


Figure 4.1: The dataset was uploaded to Roboflow

Once the dataset is uploaded, the annotation process involves manually drawing bounding boxes around cavities in the images. Each bounding box is carefully positioned to accurately outline the cavities, and appropriate class label as "caries" are assigned (Figure 4.2). The platform's annotation tools allowed for precise placement of these boxes, ensuring that each feature was accurately labeled. Additionally, the ability to apply tags to images helped in organizing the dataset further, making it easier to filter and manage the annotated data.

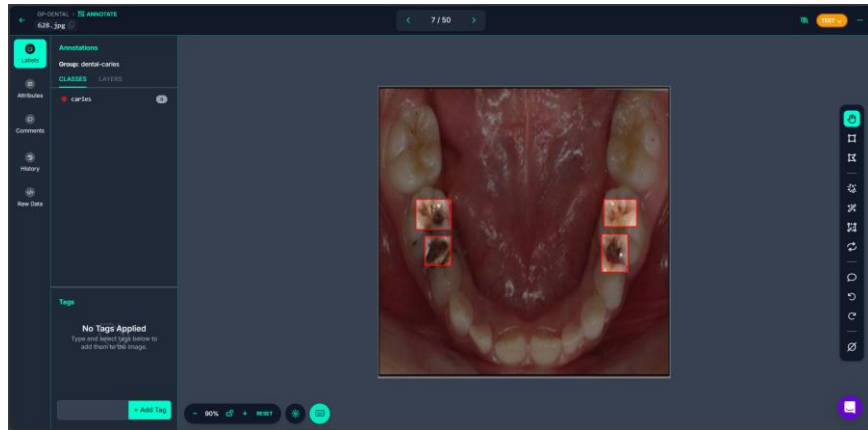


Figure 4.2: Manually drawing bounding boxes for the intraoral images

4.2.2 Data Augmentation Using Roboflow

Once annotations are complete, automatic image orientation and resizing are applied to enhance model generalization. Techniques such as image rotation, flipping, blur, and noise were employed to diversify the dataset while preserving the clinical relevance of intraoral images. These augmentations simulate real-world variability in image capture conditions without altering the anatomical or pathological features essential for diagnosis.

Augmentation parameters were carefully tuned to avoid excessive distortion that could degrade model performance. For example, while geometric transformations like rotation and horizontal flipping are useful for improving robustness to camera angle and positioning, color-based augmentations such as hue or saturation adjustments were intentionally excluded. This is because subtle color cues in intraoral images often carry diagnostic significance and altering them could introduce misleading artifacts. Thus, the selected augmentation strategy aims to balance variability with clinical fidelity.

After augmentation, the dataset was expanded to include 3807 images for training, 487 for testing, and 486 for validation, enhancing the model’s robustness and performance (Figure 4.3). The final step in the annotation workflow is exporting the dataset in a format compatible with YOLOv8. Roboflow provides an option to export augmented dataset in YOLOv8 format, which includes .txt annotation files corresponding to each image. The dataset can be downloaded for local use or accessed through a direct integration link provided by Roboflow as shown in Figure 4.4.

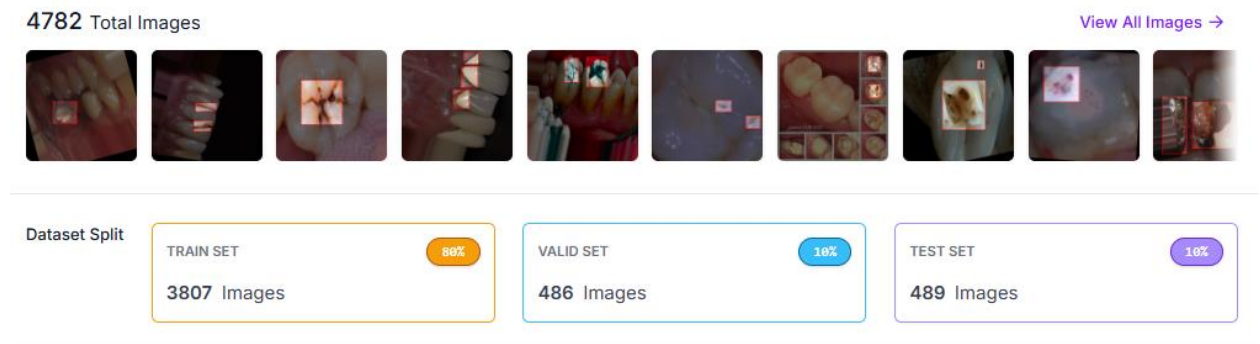


Figure 4.3: The dataset was split into three subset

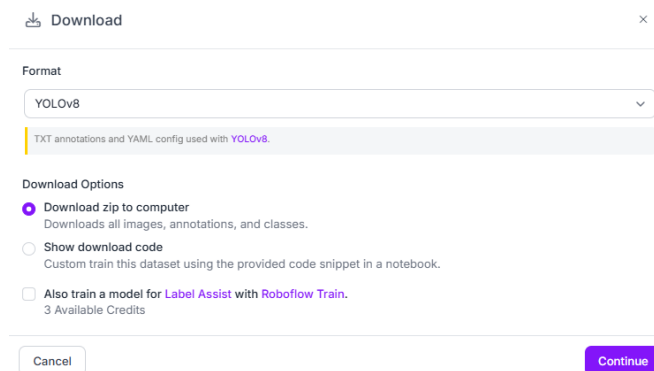


Figure 4.4: Export augmented dataset in YOLOv8 format

4.2.3 Model Training

When training the network model for dental cavity detection, the dimensions of input images were uniformly adjusted to 640×640 pixels for training and testing. This resolution enables efficient deployment on edge devices by striking a compromise between computing efficiency and maintaining essential visual characteristics. For experimentation, the PyTorch 2.4.1 deep learning framework was employed. The experimental environment featured an Intel Core i5-12500H processor, and 16 GB of RAM memory as shown in **Error! Reference source not found..** Training acceleration utilized an NVIDIA GeForce RTX 3050.

Table 4.1: The experimental environment for the model

Parameters	Configuration
CPU	I5-12500H
RAM	16GB
GPU	NVIDIA GeForce RTX 3050, 4096 MiB
GPU MEMORY	4GB
Python	3.9.2
Ultralytics	YOLOv8.2.103
DL architecture	PyTorch 2.3.1 + cu118, CUDA

The baseline version of YOLOv8n employed was Ultralytics 8.2.103. Pretraining weights were used to enhance feature extraction and improve convergence speed. The default YOLOv8n architecture in yaml file was modified by incorporating a GAM Attention module into the backbone at the final stage as shown in Figure 4.5

```
# YOLOv8.0 GAM backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9
  - [-1, 1, GAM_Attention, [1024]] # 10
```

Figure 4.5: Modified YOLOv8n backbone with the integration of the GAM Attention module

In the default implementation of YOLOv8, the `bbox_iou` function supports multiple IoU-based calculations, including GIoU, DIoU, and CIoU, but lacks of support for WiseIoU. To incorporate WiseIoU into the model, modifications are required within the `bbox_iou` function which is in `ultralytics/yolo/utils/metrics.py`. This integration involves two key enhancements. First, a new class `WIoU_Scale`, is introduced to compute a dynamic scaling

factor for IoU, thereby improving the model's sensitivity to objects of varying scales as shown in Figure 4.6. This adaptive scaling mechanism ensures that bounding box regression remains robust across different object sizes.

```

1 class WIoU_Scale:
2     ''' monotonus: {
3         None: origin v1
4         True: monotonic FM v2
5         False: non-monotonic FM v3
6     }
7     momentum: The momentum of running mean'''
8
9     iou_mean = 1.
10    monotonus = False
11    _momentum = 1 - 0.5 ** (1 / 7000)
12    _is_train = True
13
14    def __init__(self, iou):
15        self.iou = iou
16        self.update(self)
17
18    @classmethod
19    def update(cls, self):
20        if cls._is_train: cls.iou_mean = (1 - cls._momentum) * cls.iou_mean + \
21            cls._momentum * self.iou.detach().mean().item()
22
23    @classmethod
24    def _scaled_loss(cls, self, gamma=1.9, delta=3):
25        if isinstance(self.monotonus, bool):
26            if self.monotonus:
27                return (self.iou.detach() / self.iou_mean).sqrt()
28            else:
29                beta = self.iou.detach() / self.iou_mean
30                alpha = delta * torch.pow(gamma, beta - delta)
31                return beta / alpha
32        return 1

```

Figure 4.6: Adding new class for WIoU in metric python file

In addition to modifying the `bbox_iou` function, further adjustments must be made in the `BboxLoss` class within `ultralytics/yolo/Utils/loss.py` to fully integrate WiseIoU into YOLOv8's loss computation. Specifically, modifications are required in the forward function, where the existing IoU and `loss_iou` calculations must be replaced with their WiseIoU counterparts as shown in Figure 4.7. The original forward function utilizes conventional IoU-based metrics to compute localization loss, which must be adapted to incorporate WiseIoU's enhanced formulation.

```

2     iou = bbox_iou_for_nms(pred_bboxes[fg_mask], target_bboxes[fg_mask], xywh=False, WIoU
3     if type(iou) is tuple:
4         if len(iou) == 2:
5             loss_iou = ((1.0 - iou[0]) * iou[1].detach() * weight).sum() / target_scores_
6         else:
7             loss_iou = (iou[0] * iou[1] * weight).sum() / target_scores_sum
8     else:
9         loss_iou = ((1.0 - iou) * weight).sum() / target_scores_sum

```

Figure 4.7: Replacement of the iou and loss_iou functions

Initial model training was conducted using hyperparameters to validate the improvement of the proposed algorithm as shown in Table 4.2. The AdamW optimizer was utilized for a total of 120 epochs. The initial learning rate was set to 0.01. Additionally,

mosaic data augmentation techniques were employed to enhance detection capabilities. Figure 4.8 show snippet code for initial training with the hyperparameters. Patience of 10 was applied to mitigate overfitting, ensuring the model stops training if validation performance does not improve for 10 consecutive epochs.

Table 4.2: The key parameters we used during initial model training

Parameters	Setup
Epoch	120
Optimizer	AdamW
Learning rate	0.01
Image size	640
Batch size	12
Weight decay	0.0005
Momentum	0.937
Mosaic	1.0
Close mosaic	10
Confidence	0.3
Iou	0.6
Patience	10

```
import os
from ultralytics import YOLO

# 获取当前工作目录
cwd = os.getcwd()

# 数据集的yaml路径
data_yaml = r"C:\ultralytics-v8.2.103\ultralytics-ultralytics-bb4601c\gp-dental.v6i.yolov8\data.yaml"

# 训练YOLOv8模型
model = YOLO(r"C:\ultralytics-v8.2.103\ultralytics-ultralytics-bb4601c\ultralytics\cfg\models\v8\yolov8_GW_Attention.yaml").load("yolov8n.pt")
model.train(data_yaml,
            epochs=120,
            batch=12,
            patience=10,
            weight_decay=0.0005,
            momentum=0.937,
            imgsz=640,
            conf=0.3,
            iou=0.6,
            lr=0.01,
            optimizer="AdamW",
            project=os.path.join(cwd, "runs", "detect", "gp-dental.v6i.yolov8_b12e160"),
            name="yolov8n_GW_Attentionv8i10u_version") # 保存为不同版本
```

Figure 4.8: Initial model training for proposed GW-YOLOv8n

4.3 Optimization

The optimization strategy was refined to enhance model performance and generalization. Initially, the AdamW optimizer was utilized but it was later replaced with an automatic selection mechanism. The epoch was increased to 300 aimed at improving the

model's convergence and generalization. In addition, the learning rate was fine-tuned within the range of 0.01 to 0.03 to achieve an optimal balance between convergence speed and stability. Furthermore, the confidence threshold was adjusted to a range of 0.20 to 0.25 to filter out low-confidence detections while maintaining sensitivity to relevant features. Similarly, the Intersection over Union (IoU) threshold was optimized between 0.25 and 0.60 to refine object localization accuracy. Additionally, a cosine learning rate scheduler was implemented to dynamically adjust the learning rate throughout the training process, preventing stagnation and enhancing model adaptability.

Table 4.3: The modified key parameters during model optimization

Parameters	Setup
Epoch	300
Optimizer	Auto
Learning rate	0.01-0.03
Confidence	0.20-0.25
IoU	0.25-0.60
Learning rate scheduler	Cosine

This optimization aims to improve the model's robust performance in detecting dental cavity, as depicted in Figure 4.9, illustrating the training results. Table 4.4 lists the optimized hyperparameters for the best performance of the proposed model.

Table 4.4: The key parameters used for best model optimization

Parameters	Setup
Epoch	300
Optimizer	Auto
Learning rate	0.03
Image size	640
Batch size	12
Weight decay	0.0005
Momentum	0.937
Mosaic	1.0

Close mosaic	10
Confidence	0.25
IoU	0.5
patience	10
Learning rate scheduler	Cosine

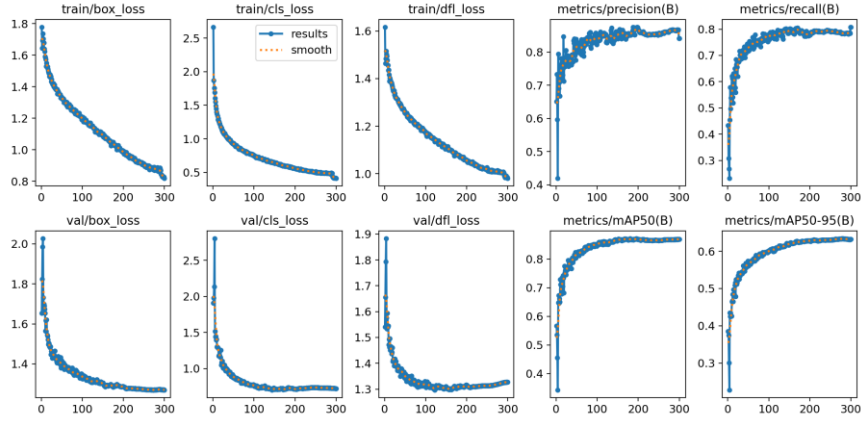


Figure 4.9: Best optimization results of the GW-YOLOv8n

4.4 Results and Analysis

Comparative studies were conducted to rigorously evaluate the superiority and effectiveness of GW-YOLOv8n. Firstly, we assess the comparison performance of the unoptimized and optimized GW-YOLOv8n model. In addition, the study involved a detailed comparison with the baseline YOLOv8n model.

4.4.1 Comparison of Fine-Tuning Results

Comparative experiments were conducted to assess the optimization performance of the GW-YOLOv8n model using both the validation and testing sets of the collected dataset. Figure 4.10 and Figure 4.11 show the snippet code for the model for validation and testing, respectively.

```

from ultralytics import YOLO
import os
# 获取当前工作目录
cwd = os.getcwd()

# 加载训练好的模型
yolov8_model = YOLO(r"C:\ultralytics-v8.2.103\ultralytics-ultralytics-bb4601c\gp-dental.v6i.yolov8_b12e160\yolov8n_GAM_Attention+MIou_version\weights\best.pt")

# 数据集的yaml路径
data_yaml = r"C:\ultralytics-v8.2.103\ultralytics-ultralytics-bb4601c\gp-dental.v6i.yolov8\data.yaml"
# 使用验证集数据进行评估
yolov8_model_result = yolov8_model.val(data=data_yaml, project=os.path.join(cwd, "runs", "validation_result", "op"),
                                       name="yolov8gam") # 数据集中定义了验证集的路径

# 打印评估结果
print("/n/yolov8_model_result/n/n")
print("yolov8_model_result")

```

Figure 4.10: Validate the optimized model using validation dataset

```

b12uf(„λoλoλ8 woqet le2n1f„)
b12uf(„u\λoλoλ8 woqet le2n1f\λu„)
# 打印结果

| | | | | u9m6= „λoλoλ8woqet le2n1f„ zbt1f = „fegf„ # 数据集集中定义的验证集路径
λoλoλ8 woqet le2n1f = λoλoλ8 woqet le2n1f λ91(q9f9=q9f9 λ9w1 b1o1ecf=02 b9f1 j01u(cmq „λu92„ „fegf1u8 le2n1f„ „0b„)
# 验证集数据集地址
q9f9 λ9w1 = „C:/n1f9jλ1c2-λ8.5.103/n1f9jλ1c2-n1f9jλ1c2-ppq00c/8b-q9f9j λ91 λoλoλ8/q9f9 λ9w1„
# 数据集的yaml路径

λoλoλ8 woqet = λoλo(„C:/n1f9jλ1c2-λ8.5.103/n1f9jλ1c2-n1f9jλ1c2-ppq00c/8b-q9f9j λ91 λoλoλ8 b12e160/λoλoλ8u cqm v1fegf1u8+MIou v6e21ou/m618m1c2/p62f bf„)
# 加载训练好的模型

cmq = 02 b9f1cmq()
# 获取当前工作目录
1wb0uf 02
1fow n1f9jλ1c2 1wb0uf λoλo

```

Figure 4.11: Testing the optimized model using testing dataset

The evaluation metrics as presented in **Error! Reference source not found.** and Table 4.6 which highlight the superior performance of the optimized GW-YOLOv8n model compared to its unoptimized counterpart in both validation and testing datasets. The optimized model demonstrates notable improvements across all key metrics and verified experimentally, as shown in Figure 4.12, demonstrating the optimized performance.

Table 4.5: Comparative analysis between optimized and unoptimized models in validation datasets

Models	Precision (%)	Recall (%)	F1-Score	mAP 0.5 (%)	mAP 0.5:0.9 (%)
GW-YOLOv8n (unoptimized)	0.839	0. 77	0.803	0. 843	0.601
GW-YOLOv8n (optimized)	0.885	0. 804	0.843	0.883	0.653

Table 4.6: Comparative analysis between optimized and unoptimized models in testing datasets

Models	Precision (%)	Recall (%)	F1-Score	mAP 0.5 (%)	mAP 0.5:0.9 (%)
GW-YOLOv8n (unoptimized)	0.834	0.778	0.805	0.851	0.603
GW-YOLOv8n (optimized)	0.887	0.799	0.841	0.879	0.659

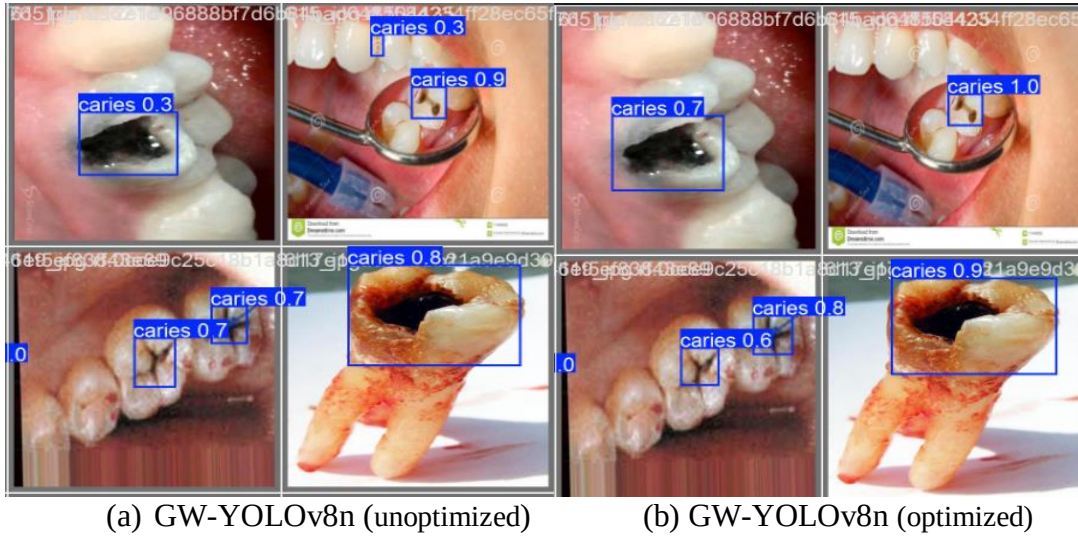


Figure 4.12: The detection outcome of unoptimized model and optimized model

Setting the optimizer to "auto" enables the model to automatically select the most appropriate optimization algorithm based on the dataset and training conditions, thereby improving convergence and learning stability. The learning rate increased to 0.03 facilitates faster training without compromising stability, ensuring that the model reaches an optimal solution more efficiently.

In addition, the confidence threshold was adjusted to 0.25 allows the model to detect a greater number of potential objects, thereby reducing the likelihood of false negatives, which is particularly beneficial for detecting small or less distinguishable objects. Additionally, setting the IoU threshold to 0.5 establishes a balanced trade-off between precision and recall, while the implementation of a cosine learning rate scheduler enables a gradual reduction in the learning rate which promotes smoother convergence by initially allowing rapid learning and subsequently fine-tuning the model with smaller updates.

4.4.2 Comparison between Model YOLOv8n and GW-YOLOv8n

Comparative experiments were conducted to evaluate the detection performance of GW-YOLOv8n against the baseline YOLOv8n model using the validation and testing set of the collected dataset. Table 4.7 and Table 4.8 presents all the evaluation metrics, demonstrating the superior performance of the GW-YOLOv8n. The effectiveness of the algorithm was also verified experimentally, as shown in Figure 4.13, demonstrating the detection performance.

Table 4.7: Comparative analysis between GW-YOLOv8n and YOLOv8n in validation dataset

Models	Precision (%)	Recall (%)	F1-Score	mAP 0.5 (%)	mAP 0.5:0.9 (%)
YOLOv8n	0.839	0.780	0.808	0.853	0.611
GW-YOLOv8n	0.885	0.804	0.843	0.883	0.653

Table 4.8: Comparative analysis between GW-YOLOv8n and YOLOv8n in testing dataset

Models	Precision (%)	Recall (%)	F1-Score	mAP 0.5 (%)	mAP 0.5:0.9 (%)
YOLOv8n	0.839	0.783	0.820	0.855	0.613
GW-YOLOv8n	0.887	0.799	0.841	0.879	0.659

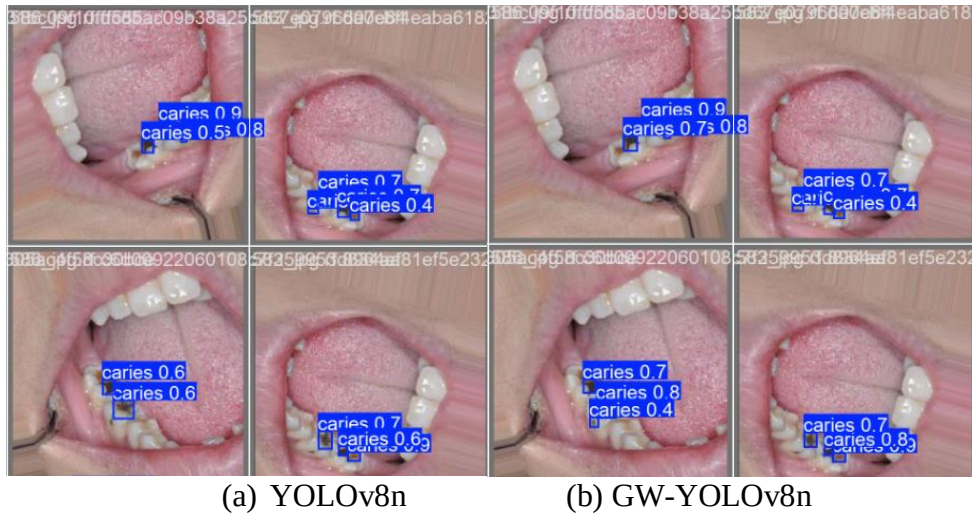


Figure 4.13: The detection outcome of proposed model and YOLOv8n

Integrating GAM attention into the backbone allows the model to learn more discriminative features, enhancing its ability to focus on important object details while

filtering out background noise. Traditional convolutional neural networks (CNNs) process all features equally, which can lead to misclassification or missed detections, especially in complex backgrounds. However, GAM attention improves feature selection by strengthening significant spatial and channel-wise information, helping the model better distinguish between objects and irrelevant regions. This improved the precision and recall, as the model becomes more effective at detecting objects with higher confidence while reducing false positives and false negatives.

At the same time, the modified WiseIoU refines bounding box regression, leading to more accurate and stable localization. Standard IoU-based loss functions primarily focus on overlapping areas but often struggle with cases where predicted bounding boxes are not perfectly aligned with ground truth boxes. The modified WiseIoU adjusts the way IoU is calculated by incorporating additional weighting mechanisms, which helps the model prioritize better localization. This modification ensures that bounding boxes are adjusted more effectively, particularly for small or overlapping objects, resulting in improved mAP scores across different IoU thresholds.

4.5 System Development

This section outlines the development of a web-based dental cavity detection system using the proposed model. The system enables users to upload intraoral images and videos for cavity detection. Additionally, real-time detection is available for immediate analysis. The development process includes system functions, architecture, and implementation steps.

The frontend of the web-based dental cavity detection system was developed using React.js, along with HTML, CSS, and JavaScript to ensure a responsive and user-friendly interface. The backend is powered by FastAPI, providing a fast and efficient framework for handling data processing and API requests.

4.5.1 Home Page

The homepage of the web-based dental cavity detection system serves as an educational hub, providing information on the causes, prevention, and treatment of cavities. Designed to be user-

friendly and visually engaging, it helps both dental professionals and patients understand oral health better. The homepage features interactive resources, informative content, and easy navigation to enhance the learning experience. Figure 4.14 illustrates the layout and key elements of the homepage while Figure 4.15 shows the code implementation for the homepage.



Figure 4.14: Homepage for the dental cavity detection system

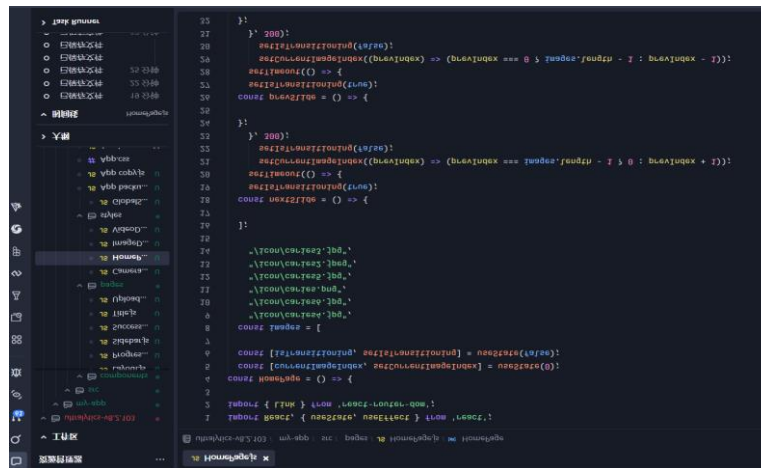


Figure 4.15: Implementation of homepage using React.js

4.5.2 Image Detection

The image detection page of the web-based dental cavity detection system is the core interface where users can upload intraoral images for analysis as shown in Figure 4.16. This page is designed to be user-friendly and efficient, allowing both dental professionals and patients to easily submit images and receive detection results.

Once an image is uploaded, the system processes it using proposed models to identify potential cavities. The results are then displayed in a visually intuitive manner, highlighting affected areas with bounding box annotations and the result of processed image can be downloaded.

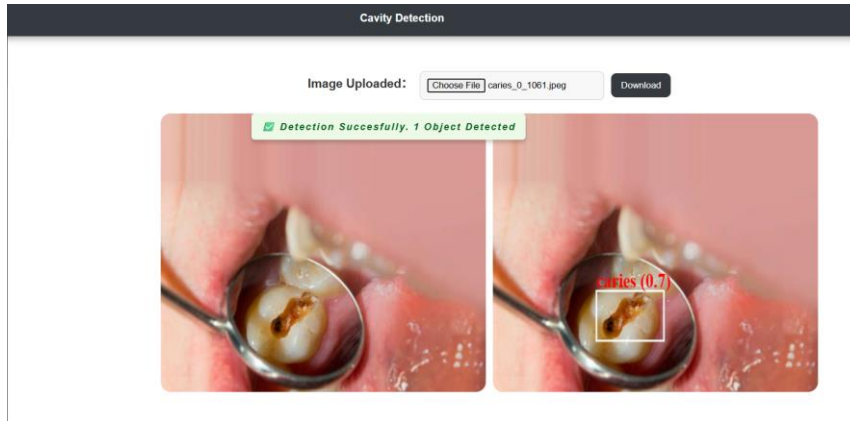


Figure 4.16: Cavity image detection results using the YOLOv8 model

The front end allows users to upload intraoral images through an intuitive interface. Once an image is selected, it is converted into a FormData object and sent via a POST request to the back end (Figure 4.17). The system then waits for the response containing detection results, which are used to overlay bounding boxes and labels on the image to highlight detected cavities.

```

object_detector.py  JS ImageDetection.js x
ultralytics-v8.2.103 / my-app / src / pages / JS ImageDetection.js / ...
8  const ImageDetection = () => {
45
46      const handleFileChange = async (event) => {
47          const selectedFile = event.target.files[0];
48          if (!selectedFile) return;
49
50          setFile(selectedFile);
51          setShowSuccessMessage(true);
52
53          const data = new FormData();
54          data.append('image_file', selectedFile, 'image_file');
55
56          try {
57              const response = await fetch(`${API_BASE_URL}/detect`, {
58                  method: 'POST',
59                  body: data
60              });
61
62              if (!response.ok) throw new Error('Failed to upload file');
63
64              const boxes = await response.json();
65              drawImageAndBoxes(selectedFile, boxes);
66
67          } catch (error) {
68              console.error('Failed to upload file:', error);
69              alert('Failed to upload file. Please try again...');
70          }
71      };
72

```

Figure 4.17: Front end sent POST request to process image detection

On the back end, FastAPI handles the image upload and passes the received file to the detection model as shown in Figure 4.18. The model processes the image and predicts the presence of cavities by generating bounding boxes with confidence scores. The results are formatted as a JSON array containing coordinates, object class, and probability values before being sent back to the front end (Figure 4.19).

```

138 @app.route("/detect", methods=["POST"])
139 def detect():
140     """
141     Handler of /detect POST endpoint
142     Receives uploaded file with a name "image_file", passes it
143     through YOLOv8 object detection network and returns an array
144     of bounding boxes.
145     :return: a JSON array of objects bounding boxes in format [[x1,y1,x2,y2,object_type,probability],...]
146     """
147     buf = request.files["image_file"]
148     print(buf) # 打印接收到的图像数据
149     boxes = detect_objects_on_image(buf.stream)
150     print(boxes) # 打印返回的边界框数据
151     return jsonify(boxes)

```

Figure 4.18: The backend FastAPI implementation for handling image detection

```

153 def detect_objects_on_image(buf):
154     """
155     Function receives an image,
156     passes it through YOLOv8 neural network
157     and returns an array of detected objects
158     and their bounding boxes
159     :param buf: Input image file stream
160     :return: Array of bounding boxes in format [[x1,y1,x2,y2,object_type,probability],...]
161     """
162     model = YOLO(r"C:\ultralitics-v8.2.103\ultralitics-ultralitics-bb4601c\runs\detect\gp-de
163     results = model.predict(Image.open(buf), iou=0.45, conf=0.48, imgsz=640)
164     result = results[0]
165     output = []
166     for box in result.boxes:
167         x1, y1, x2, y2 = [
168             round(x) for x in box.xyxy[0].tolist()
169         ]
170         class_id = box.cls[0].item()
171         prob = round(box.conf[0].item(), 2)
172         output.append([
173             x1, y1, x2, y2, result.names[class_id], prob
174         ])
175     return output

```

Figure 4.19: Function used to predict uploaded image using proposed model

4.5.3 Video Detection

The video detection page of the web-based dental cavity detection system allows users to upload intraoral videos for analysis (refer to Figure 4.20). Once a video is uploaded, the system begins analysing each frame using the proposed models. During processing, a progress bar is displayed to indicate the status, ensuring users can track the completion time. Once the analysis is finished, the system generates a processed video with bounding box annotations highlighting the detected cavities. The final processed video is then available for users to download.

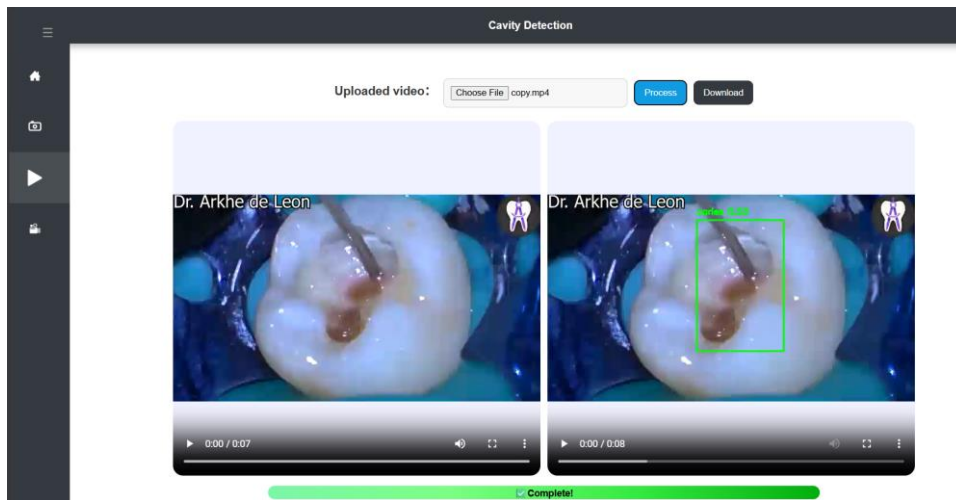


Figure 4.20: Processed video output with detected cavities highlighted

The front end allows users to upload intraoral videos through an intuitive interface. Once a video is selected, it is converted into a FormData object and sent via a POST request to the back end. The system then waits for the response containing detection results as shown in Figure 4.21. On the back end, FastAPI handles the video upload and forwards the received file to the detection model, as shown in Figure 4.22. The model processes the video frame by frame, applying the detection model to identify cavities. Bounding boxes and confidence scores are generated for each frame, and the processed frames are compiled into a final video output. The results are formatted as a JSON array containing bounding box coordinates, object class, and probability values, which are then used to overlay annotations onto the video.

```

T30 }
T31
T32 def process_video(video_path):
T33     """
T34     Process the video frame by frame, applying the detection model to
T35     identify cavities. Bounding boxes and confidence scores are generated
T36     for each frame, and the processed frames are compiled into a final
T37     video output. The results are formatted as a JSON array containing
T38     bounding box coordinates, object class, and probability values, which
T39     are then used to overlay annotations onto the video.
T40     """
T41     # Load the video
T42     video = cv2.VideoCapture(video_path)
T43     # Get the video properties
T44     width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
T45     height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
T46     # Create a list to store the bounding boxes and confidence scores
T47     results = []
T48     # Process the video frame by frame
T49     while video.isOpened():
T50         # Read the next frame
T51         ret, frame = video.read()
T52         if not ret:
T53             break
T54         # Apply the detection model to the frame
T55         results.append(detect_cavities(frame))
T56     # Release the video capture object
T57     video.release()
T58     # Return the results
T59     return results

```

Figure 4.21: Code implementation sent POST request to process uploaded video

```

589  f0t9f"tjwme = tuf(cab'det(cas'cvb'b80b'ebvme'conm1)) # 检测视频
590  tba = tuf(cab'det(cas'cvb'b80b'eb8))
591  p0t9f = tuf(cab'det(cas'cvb'b80b'ebvme'height))
592  m1qf = tuf(cab'det(cas'cvb'b80b'ebvme'mid1h))
593  cab = cas'atq00c0b1n0e(tjwme)
594  w0q0f = a0f0(l-c://n1f0b1f1t02-a8's't02//n1f0b1f1t02-n1f0b1f1t02-p0q0q0c//u1u2//q0e0c0f/
595
596  b10d0e22"q0f0[,"b10d0e22"] = 0 # 检测进度
597  2j0n1q"2f0b = t0f0e # 检测进度
598  t0"b10c0e22j0d = t1ue
599
600  t1f0"2010(tjwme)
601
602  cou1e1e0"t1f0wme = 02"b0f0"j0t0(b10c0e22e0"e0f0e0" t1"b10c0e22e0"t1x0q"(t1f0"t1f0wme))
603  b10c0e22e0"t1f0wme = 02"b0f0"j0t0(b10c0e22e0"e0f0e0" t1"b10c0e22e0"t1f0"t1f0wme))
604  t1f0wme = 02"b0f0"j0t0(b10c0e22e0"e0f0e0" t1f0"t1f0wme)
605
606  t1 t1f0"t1f0wme = ,:~
607  t1f0 = 10d0e22"t1f0e[,"t1q00"t1f0,]
608
609  t1 ,t1q00"t1f0, u0f t1 10d0e22"t1f0e:~
610
611  t1 t0"b10c0e22j0d:~
612
613  0f0p0f b10d0e22"q0f0" t0"b10c0e22j0d" 2j0n1q"2f0b" 0
614  q0t b10c0e22"t1q00":
615  @0b0"10n1e(,"b10c0e22"t1q00" w0f0p0e["b021,])

```

Figure 4.22: Main function to process detection on uploaded video

The progress bar is dynamically updated throughout the detection process to provide real-time feedback on the video's analysis status as shown in Figure 4.23. This is implemented by sending POST requests every second in front end to update the progress status (Figure 4.24). Once the analysis is complete, the progress bar reaches 100%, and the detected results are displayed alongside the original video. This feature enhances user experience by keeping them informed and engaged throughout the detection process.

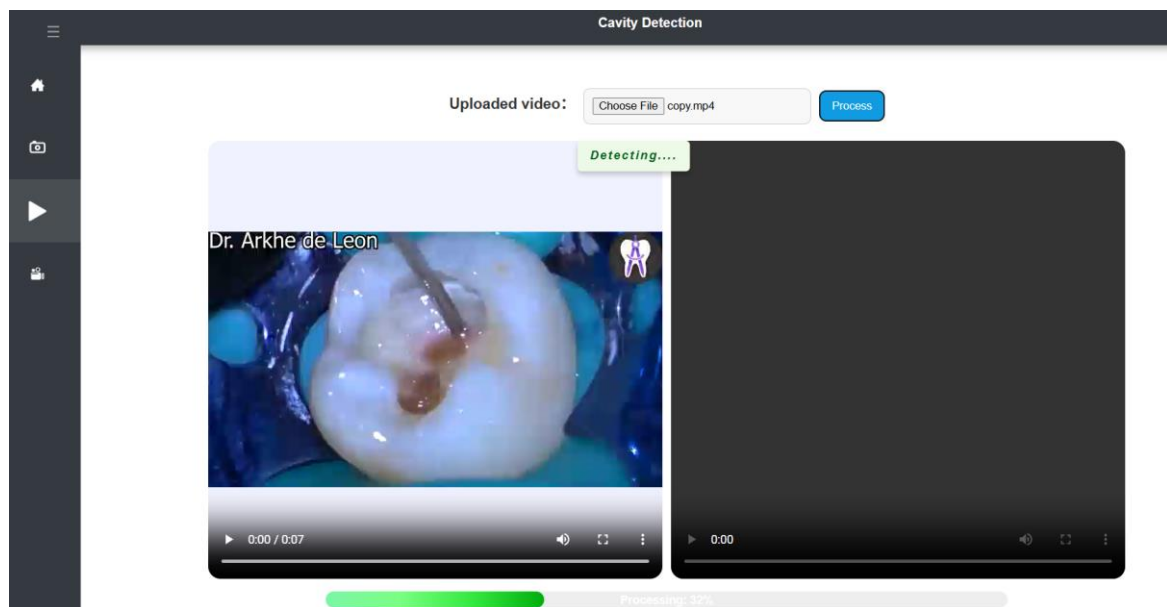


Figure 4.23: Progress bar was displayed during processing video detection

```

225 // 更新进度
226 const updateProgress = async () => {
227   try {
228     const response = await fetch(`${API_BASE_URL}/progress`);
229     const data = await response.json();
230     console.log(data.progress);
231
232     setProgress(data.progress);
233     setProgressText(`Processing: ${data.progress}%`);
234     console.log('processing:', isProcessing);
235     if (data.progress < 100 && isProcessing) {
236       setTimeout(updateProgress, 1000); // 继续请求进度
237     } else if (data.progress >= 100) {
238       setProgressText('✅ Complete!');
239     }
240   } catch (error) {
241     console.error('Failed to update progress:', error);
242     if (isProcessing) {
243       setTimeout(updateProgress, 1000); // 出错后继续尝试
244     }
245   }
246 };
247

```

Figure 4.24: POST request every second to update progress of video detection

4.5.4 Real Time Detection

The real-time detection module enables users to perform live cavity detection using an external camera. This feature is designed for dental professionals and researchers who require instant analysis without the need for pre-recorded videos. Once the camera feed is activated, frames are continuously captured and processed by the detection model in real time. The system overlays bounding boxes and labels on detected cavities, providing immediate visual feedback on the live video stream. Figure 4.25 illustrate real-time detection modules.

The real-time detection module optimizes frame processing speed while maintaining high detection accuracy to ensure smooth performance. A buffering mechanism is implemented to handle varying frame rates, preventing lag and ensuring a seamless user experience.

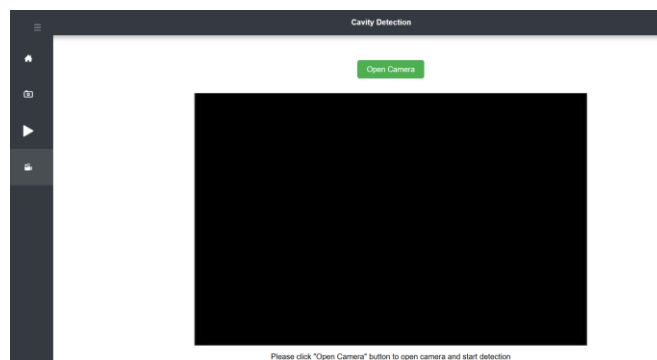


Figure 4.25: Real time detection for the proposed model

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Introduction

This chapter provides a detailed and structured summary of the principal achievements attained throughout the course of the project. It revisits the initial objectives articulated at the outset of the study and assesses the degree to which these objectives have been successfully realized. In addition, this chapter offers a critical analysis of the limitations encountered during the project's development, including constraints related to data availability, model performance, and generalizability. These limitations are examined with the aim of identifying specific areas requiring further refinement. Finally, the chapter outlines prospective directions for future research and enhancement, suggesting possible strategies to augment the system's accuracy, efficiency, and clinical applicability. By consolidating the insights gained, this chapter seeks to contribute to the ongoing advancement of artificial intelligence applications in dental diagnostics and to lay the groundwork for subsequent research in this domain.

5.2 Achievements

Table 5.1: Objectives and achievements of the proposed dental caries detection system

Objectives	Achievements
To investigate computer vision and deep learning approaches for cavity detection.	Successfully researched and adapted a modified YOLOv8 model capable of detecting caries in non-X-ray images.
To develop a system using image processing and deep learning techniques for detecting dental caries	Developed and trained a modified YOLOv8-based detection system that can accurately identify carious lesions across multiple teeth in a single non-specialized image.
To evaluate the performance of the proposed system in terms of detection accuracy.	Evaluated the system using key performance metrics such as accuracy, precision, recall, F1 score, confirming reliable detection across varying image qualities and lighting conditions.

Table 5.1 outlines the key research objectives and corresponding outcomes achieved in the development of an automated dental caries detection system leveraging computer vision and deep learning methodologies. The initial objective involved a comprehensive investigation into state-of-the-art object detection frameworks, culminating in the successful adaptation and modification of the YOLOv8 architecture. This customized model was specifically optimized to detect dental carious lesions in non-radiographic intraoral images, thereby extending its applicability to scenarios where specialized imaging equipment may not be available.

Subsequently, a complete detection system was developed, incorporating both advanced image preprocessing techniques and the modified YOLOv8 model. The system demonstrated the capability to accurately localize and classify carious lesions across multiple teeth within a single intraoral image, maintaining robust performance across diverse image resolutions, lighting conditions, and patient-specific anatomical variations.

Finally, the system was subjected to a comprehensive performance evaluation utilizing a meticulously curated dataset. Quantitative assessment of detection efficacy was conducted through the application of standard performance metrics, including accuracy, precision, recall, and F1-score. The experimental findings demonstrated that the proposed system consistently delivers reliable detection performance, underscoring its potential as a non-invasive and accessible solution for preliminary screening and diagnosis of dental caries.

5.3 Limitations

Despite the project achieving its primary objectives and demonstrated encouraging results in the field of automated dental cavity detection, several limitations emerged throughout its development that warrant critical consideration. Firstly, the variability in intraoral image quality presented a considerable challenge. Factors such as inconsistent lighting conditions, occlusions, image blur, low contrast, and suboptimal angles during image acquisition significantly influenced the model's ability to accurately detect cavities. In clinical practice, images are rarely captured under standardized conditions, and this real-world variability can lead to reduced model reliability when applied outside of a controlled experimental environment.

Secondly, the variability in intraoral image quality presents a considerable challenge. Factors such as inconsistent lighting conditions, occlusions, image blur, low contrast, and suboptimal angles during image acquisition significantly influenced the model's ability to accurately detect cavities. In clinical practice, images are rarely captured under standardized conditions, and this real-world variability can lead to reduced model reliability when applied outside of a controlled experimental environment.

An additional significant limitation is associated with the use of the modified YOLOv8 object detection model. While the model demonstrates notable efficiency and satisfactory accuracy, it remains vulnerable to specific types of prediction errors. In particular, the model may fail to detect early-stage or visually indistinct carious lesions, thereby resulting in false negatives. Conversely, benign anatomical variations or harmless pigmentation within the oral cavity may be erroneously classified as dental caries, leading to false positives. Such misclassifications, if not addressed adequately, have the potential to undermine the diagnostic accuracy of the system and diminish user confidence in its reliability.

In addition, the absence of clinically validated, expert-annotated labels for all training samples introduces a degree of uncertainty in the ground truth used for supervised learning. The quality and reliability of model predictions are inherently tied to the accuracy of the labels, and inconsistencies in annotation can adversely impact the model's learning process and evaluation metrics.

Lastly, computational resource constraints limited the exploration of more complex deep learning architectures that might offer improved performance. Advanced models such as transformer-based networks or ensembles could potentially yield higher accuracy but were not implemented due to hardware limitations and training time constraints.¹

5.4 Future Works

Building upon the outcomes and insights gained from this study, several recommendations are proposed to guide future research and development efforts aimed at enhancing the performance,

usability, and clinical relevance of the dental cavity detection system. These prospective directions are intended to address the identified limitations and to expand the scope and impact of the current work.

Firstly, one of the most critical areas for future improvement is the expansion and diversification of the dataset. Incorporating a larger volume of high-quality, clinically annotated intraoral images from a wide range of demographic groups and dental conditions would significantly improve the model's generalization capabilities. Collaboration with dental institutions and professionals to obtain verified clinical data will also enhance the accuracy and validity of the training labels, thereby strengthening the model's reliability. Efforts should be made to improve the system's robustness to real-world imaging conditions. This may include incorporating advanced image enhancement techniques or developing preprocessing algorithms specifically designed to handle issues such as poor lighting, motion blur, and occlusion. These enhancements would allow the model to perform more consistently in diverse clinical and non-clinical settings.

Secondly, exploring more advanced deep learning architectures such as transformer-based models, attention mechanisms, or ensemble learning approaches may further increase detection accuracy and reduce error rates. These models, although more computationally intensive, have demonstrated superior performance in other domains and hold promises for medical imaging tasks as well.

Another important direction involves the integration of multimodal data sources. Combining image data with additional contextual information such as patient history, age, oral hygiene habits, and previous diagnoses could support more comprehensive and personalized diagnostic outputs. Such integration would not only improve accuracy but also align the system more closely with the decision-making processes used by dental professionals. Furthermore, it is recommended that the system undergo rigorous clinical validation through pilot studies in real dental practice settings. User feedback from both clinicians and patients should be systematically collected to refine the system's user interface, improve interpretability of results, and ensure compliance with ethical and regulatory standards in healthcare technologies.

Lastly, the deployment of the system in the form of a scalable web or mobile application should be further developed to ensure accessibility for a broader range of users, including individuals in remote or underserved areas. Features such as multilingual support, offline functionality, and real-time feedback could be introduced to improve user experience and broaden the system's impact.

References

- Albahli, S., Ayub, N., & Shiraz, M. (2021). Coronavirus disease (COVID-19) detection using X-ray images and enhanced DenseNet. *Applied Soft Computing*, 107645. <https://doi.org/10.1016/j.asoc.2021.107645>
- Albano, D., Galiano, V., Basile, M., Filippo Di Luca, Gitto, S., Messina, C., Maria Grazia Cagetti, Massimo Del Fabbro, Gianluca Martino Tartaglia, & Luca Maria Sconfienza. (2024). Artificial intelligence for radiographic imaging detection of caries lesions: a systematic review. *BMC Oral Health (Online)*, 24(1). <https://doi.org/10.1186/s12903-024-04046-7>
- AlSayed, A., Taqateq, A. M., Al-Sayed, R., Suleiman, D., Shukri, S., Alhenawi, E., & Albsheish, A. M. (2023). Employing CNN ensemble models in classifying dental caries using oral photographs. *International Journal of Data and Network Science*, 7(4), 1535–1550. doi:10.5267/j.ijdns.2023.8.009
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1). <https://doi.org/10.1186/s40537-021-00444-8>
- Angelone, F., Ponsiglione, A. M., Ricciardi, C., Cesarelli, G., Sansone, M., & Amato, F. (2023). Diagnostic Applications of Intraoral Scanners: A Systematic Review. *Journal of Imaging*, 9(7), 134. <https://doi.org/10.3390/jimaging9070134>
- Arabi, M. Y., Ramadan, G. M., Reddy, R. A., Habelalmateen, M. I., & Priyanka K. (2023). *Dental X-Ray Image Analysis for Diagnosis Utilized Convolutional Neural Network Using Real-Time Performance*. 1–6. <https://doi.org/10.1109/icmnwc60182.2023.10435896>

- BabaAhmadi, A., Masoumi, F. S., Ayati, M., & AmirPour, M. (2024, February 21). A transfer-learning-based strategy for autonomous driving: Leveraging driver experience for vision-based control. *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, 1–6. Presented at the 2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP), Babol, Iran, Islamic Republic of. doi:10.1109/aisp61396.2024.10475214
- Bernabé, E., Masood, M., & Vujicic, M. (2017). The impact of out-of-pocket payments for dental care on household finances in low and middle income countries. *BMC Public Health*, 17(1). <https://doi.org/10.1186/s12889-017-4042-0>
- Bhargava, N., Rathore, P. S., Vaishnav, P., & Goswami, A. (2023). *A Novel Single Shot Multibox Detector based Face Mask Detection*. 1028–1031. <https://doi.org/10.1109/icaiss58487.2023.10250518>
- Cai, Y., Zhang, X., Kovalsky, S. Z., Ghashghaei, H. T., & Greenbaum, A. (2021). Detection and classification of neurons and glial cells in the MADM mouse brain using RetinaNet. *PLOS ONE*, 16(9), e0257426. <https://doi.org/10.1371/journal.pone.0257426>
- Centers for Disease Control and Prevention. (2024, May 10). *Health Disparities in Oral Health*. Oral Health. <https://www.cdc.gov/oral-health/health-equity/index.html>
- Chollet, F. (2016). Xception: Deep Learning with Depthwise Separable Convolutions. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1610.02357>
- Coskun, M., Ucar, A., Yildirim, O., & Demir, Y. (2017). Face recognition based on convolutional neural network. *2017 International Conference on Modern Electrical and Energy Systems (MEES)*. <https://doi.org/10.1109/mees.2017.8248937>

- Dabwan, B. A., Olayah, F. A., Halawani, H. T., Mashraqi, A. M., Abdelrahman, Y. A., & Shamsan, A. F. (2023). *Classification of Sign language Using VGG16 and VGG19*. 1–6. <https://doi.org/10.1109/isas60782.2023.10391673>
- Di, J., Ren, L., Liu, J., Guo, W., Huaikun Zhange, Liu, Q., & Lian, J. (2023). FDNet: An end-to-end fusion decomposition network for infrared and visible images. *PloS One*, 18(9), e0290231–e0290231. <https://doi.org/10.1371/journal.pone.0290231>
- Doğan, Y. (2024). Which pooling method is better: Max, Avg, or Concat (Max, Avg). *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, 66(1), 95–117. <https://doi.org/10.33769/aupse.1356138>
- Duhan, S., Preeti Gulia, Gill, N. S., Shukla, P. K., Khan, S. B., Ahlam Almusharraf, & Alkhaldi, N. (2024). Investigating attention mechanisms for plant disease identification in challenging environments. *Heliyon*, 10(9), e29802–e29802. <https://doi.org/10.1016/j.heliyon.2024.e29802>
- Guo, H., Huang, S., Huang, C., Shi, F., Zhang, M., & Pan, Z. (2020). Binary File's Visualization and Entropy Features Analysis Combined with Multiple Deep Learning Networks for Malware Classification. *Security and Communication Networks*, 2020, e8881760. <https://doi.org/10.1155/2020/8881760>
- Hu, X., Chu, L., Pei, J., Liu, W., & Bian, J. (2021). Model Complexity of Deep Learning: A Survey. *ArXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2103.05127>
- Hu, Z., Zhang, J., & Ge, Y. (2021). Handling Vanishing Gradient Problem Using Artificial Derivative. *IEEE Access*, 9, 22371–22377. <https://doi.org/10.1109/access.2021.3054915>
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269. <https://doi.org/10.1109/cvpr.2017.243>

- Hussain, M. (2024, July 3). *YOLOv5, YOLOv8 and YOLOv10: The Go-To Detectors for Real-time Vision*. ArXiv.org. <https://doi.org/10.48550/arXiv.2407.02988>
- Hussain, M., & Khanam, R. (2024). In-Depth Review of YOLOv1 to YOLOv10 Variants for Enhanced Photovoltaic Defect Detection. *Solar*, 4(3), 351–386.
<https://doi.org/10.3390/solar4030016>
- Jagtap, A., & Karniadakis, G. E. (2023). How important are activation functions in regression and classification? A survey, performance comparison, and future directions. *Journal of Machine Learning for Modeling and Computing*.
<https://doi.org/10.1615/jmachlearnmodelcomput.2023047367>
- Juyal, A., Tiwari, H., Singh, U. K., Kumar, N., & Kumar, S. (2023). Dental Caries Detection Using Faster R-CNN and YOLO V3. *ITM Web of Conferences*, 53, 02005.
<https://doi.org/10.1051/itmconf/20235302005>
- Khanam, R., Hussain, M., Hill, R., & Allen, P. (2024). A Comprehensive Review of Convolutional Neural Networks for Defect Detection in Industrial Applications. *IEEE Access*, 1–1. <https://doi.org/10.1109/access.2024.3425166>
- Krichen, M. (2023). Convolutional Neural Networks: A Survey. *Computers*, 12(8), 151–151.
<https://doi.org/10.3390/computers12080151>
- Laishram, A., & Thongam, K. (2020, February 1). *Detection and Classification of Dental Pathologies using Faster-RCNN in Orthopantomogram Radiography Image*. IEEE Xplore. <https://doi.org/10.1109/SPIN48934.2020.9071242>
- Lanham, M. (2021). Attention Is All We Need! *Apress EBooks*, 195–222.
https://doi.org/10.1007/978-1-4842-7092-9_7
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016*, 9905, 21–37.
https://doi.org/10.1007/978-3-319-46448-0_2

- Lysechko, V. P., Sadovnykov, B. I., Komar, O. M., & Zhuchenko, O. S. (2024). A RESEARCH OF THE LATEST APPROACHES TO VISUAL IMAGE RECOGNITION AND CLASSIFICATION. *Radìoelektronìka, Ìnformatika, Upravlìnnâ/Radìoelektronika, Ìnformatika, Upravlìnnâ, 1*, 140–140.
<https://doi.org/10.15588/1607-3274-2024-1-13>
- Mahoor, M. H., & Abdel-Mottaleb, M. (2005). *Automatic classification of teeth in bitewing dental images*. <https://doi.org/10.1109/icip.2004.1421863>
- Maji, P., & Mullins, R. (2018). On the Reduction of Computational Complexity of Deep Convolutional Neural Networks. *Entropy*, 20(4), 305.
<https://doi.org/10.3390/e20040305>
- Manakitsa, N., Maraslidis, G. S., Moysis, L., & Fragulis, G. F. (2024). A Review of Machine Learning and Deep Learning for Object Detection, Semantic Segmentation, and Human Action Recognition in Machine and Robotic Vision. *Technologies*, 12(2), 15.
<https://doi.org/10.3390/technologies12020015>
- Mehrish, A., Majumder, N., Bharadwaj, R., Mihalcea, R., & Poria, S. (2023). A review of deep learning techniques for speech processing. *Information Fusion*, 101869.
<https://doi.org/10.1016/j.inffus.2023.101869>
- Michou, S., Lambach, M. S., Ntovas, P., Benetti, A. R., Bakhshandeh, A., Rahiotis, C., Ekstrand, K. R., & Vannahme, C. (2021). Automated caries detection in vivo using a 3D intraoral scanner. *Scientific Reports*, 11(1). <https://doi.org/10.1038/s41598-021-00259-w>
- Ng, K. C. (2024). AI in Paediatric Radiology. *Journal of Medical Imaging and Radiation Sciences*, 55(3), 101459–101459. <https://doi.org/10.1016/j.jmir.2024.101459>
- Ogundokun, R. O., Li, A., Babatunde, R. S., Umezuruike, C., Sadiku, P. O., Abdulahi, A. T., & Babatunde, A. N. (2023). Enhancing Skin Cancer Detection and Classification in

- Dermoscopic Images through Concatenated MobileNetV2 and Xception Models. *Bioengineering*, 10(8), 979. <https://doi.org/10.3390/bioengineering10080979>
- Bastewad, P. (2024). *Dental Caries Dataset*. Kaggle.com. <https://www.kaggle.com/datasets/prajwalbastewad/dental-caries-dataset>
- Prinzi, F., Insalaco, M., Orlando, A., Gaglio, S., & Vitabile, S. (2023). A Yolo-Based Model for Breast Cancer Detection in Mammograms. *Cognitive Computation*, 16(1), 107–120. <https://doi.org/10.1007/s12559-023-10189-6>
- Razaghi, M., Komleh, H. E., Dehghani, F., & Shahidi, Z. (2024, March 6). Innovative diagnosis of dental diseases using YOLO V8 deep learning model. *2024 13th Iranian/3rd International Machine Vision and Image Processing Conference (MVIP)*. Presented at the 2024 13th Iranian/3rd International Machine Vision and Image Processing Conference (MVIP), Tehran, Iran, Islamic Republic of. [doi:10.1109/mvip62238.2024.10491172](https://doi.org/10.1109/mvip62238.2024.10491172)
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. <https://doi.org/10.1109/cvpr.2016.91>
- Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *Arxiv.org*. <https://doi.org/10.48550/arXiv.1804.02767>
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149. <https://doi.org/10.1109/tpami.2016.2577031>
- Sarker, I. H. (2021). Deep Learning: a Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, 2(6). Springer. <https://doi.org/10.1007/s42979-021-00815-1>

- Shi, C., Lei, M., You, W., Ye, H., & Sun, H. (2024). Enhanced Floating Debris Detection Algorithm Based on CDW-YOLOv8. *Physica Scripta*. <https://doi.org/10.1088/1402-4896/ad5657>
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1).
<https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0>
- Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Science*. <https://doi.org/10.48550/arXiv.1409.1556>
- Sonavane, A., Yadav, R., & Khamparia, A. (2021). Dental cavity Classification of using Convolutional Neural Network. *IOP Conference Series: Materials Science and Engineering*, 1022, 012116. <https://doi.org/10.1088/1757-899x/1022/1/012116>
- Song, C., Jeong, Y., Huh, H., Park, J.-W., Paeng, J.-Y., Ahn, J., Son, J., & Jung, E. (2024). Multi-Scale 3D Cephalometric Landmark Detection Based on Direct Regression with 3D CNN Architectures. *Diagnostics*, 14(22), 2605.
<https://doi.org/10.3390/diagnostics14222605>
- Taye, M. M. (2023). Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. *Computation*, 11(3), 52.
<https://doi.org/10.3390/computation11030052>
- Taye, M. M. (2023). Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Computers*, 12(5), 91–91. Mdpi.
<https://doi.org/10.3390/computers12050091>
- Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, 5(4), 1680–1716.
<https://doi.org/10.3390/make5040083>

- Thanh, M. T. G., Van Toan, N., Ngoc, V. T. N., Tra, N. T., Giap, C. N., & Nguyen, D. M. (2022). Deep Learning Application in Dental Caries Detection Using Intraoral Photos Taken by Smartphones. *Applied Sciences*, 12(11), 5504. <https://doi.org/10.3390/app12115504>
- Vig, S., Arora, A., & Arya, G. (2022). Automated License Plate Detection and Recognition Using Deep Learning. 419–431. https://doi.org/10.1007/978-3-031-23724-9_39
- Vujicic, M., Buchmueller, T., & Klein, R. (2016). Dental Care Presents The Highest Level Of Financial Barriers, Compared To Other Types Of Health Care Services. *Health Affairs*, 35(12), 2176–2182. <https://doi.org/10.1377/hlthaff.2016.0800>
- Wen, P. Y. F., Chen, M. X., Zhong, Y. J., Dong, Q. Q., & Wong, H. M. (2021). Global Burden and Inequality of Dental Caries, 1990 to 2019. *Journal of Dental Research*, 101(4), 002203452110562. <https://doi.org/10.1177/00220345211056247>
- World Health Organization. (2024, November 6). *Oral health*. World Health Organization. <https://www.who.int/news-room/fact-sheets/detail/oral-health>
- Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional Neural networks: an Overview and Application in Radiology. *Insights into Imaging*, 9(4), 611–629. <https://doi.org/10.1007/s13244-018-0639-9>
- Yi, L., Junjie, S., Buyun, W., Yongde, Z., & Yong, X. (2024). Research on puncture classification method of seed implantation robot based on force/image model. *Advances in Mechanical Engineering*, 16(6). doi:10.1177/16878132241258826
- Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., Dutta, A. K., & Almotairi, S. (2022). A Comparison of Pooling Methods for Convolutional Neural Networks. *Applied Sciences*, 12(17), 8643. <https://doi.org/10.3390/app12178643>

Zebari, N. A., Marqas, R. B., Salih, M. S., & Alkurdi, A. A. H. (2023). Enhancing Brain Tumor Classification with Data Augmentation and DenseNet121. *Academic Journal of Nawroz University*, 12(4), 323–334. <https://doi.org/10.25007/ajnu.v12n4a1985>