Journal of Information Hiding and Multimedia Signal Processing ©2024 ISSN 2073-4212
Ubiquitous International Volume **15**, Number **4**, December **2024**

Detection of Dental Decay in children using Modern Deep Learning Models

Tuan Nguyen Huu

Posts and Telecommunications Institute of Technology
Ha Dong, Ha Noi, Viet Nam
huutuan1705@gmail.com

Quynh Dao Thi Thuy

Posts and Telecommunications Institute of Technology
Ha Dong, Ha Noi, Viet Nam
quynhdtt@ptit.edu.vn

*Corresponding author: Quynh Dao Thi Thuy

Abstract. *Dental decay is one of the most common oral health issues worldwide. Direct examination methods by dentists with digital imaging remain the top priority. However, with the advancements in Artificial Intelligence (AI), particularly in the field of Computer Vision, major progress has been made, allowing for the automated detection of dental decay. We conduct a comparative research on three modern deep learning models for object detection: Faster R-CNN, YOLO, and DETR, applied on a dental decay dataset that we independently collected. Experiments show that modern deep learning models can meet the requirements for detecting dental decay in standardized dental images. However, these models still require further improvements in the future.*
Keywords: Detect decay, Deep learning, Object detection, Health Care

1. **INTRODUCTION.** Recording oral health status, detecting, and diagnosing conditions, including dental decay, are routine tasks in dentistry. These diagnoses provide patients with advice on disease prevention and treatment [1]. From a clinical perspective, direct examination is the preferred method due to the fact that it can be performed easily and achieves acceptable accuracy after oral cleaning [2][3][4].

However, even experienced dentists can provide conflicting diagnoses. Therefore, independent verification through methods that use artificial intelligence can meet the desired requirements. Although visual examination (VE) remains the preferred approach for identifying dental decay, analyzing digital images of the oral cavity in a machine-readable format can serve as an equivalent method. These images meet the essential criteria needed for automated analysis.

The first published papers proposed using convolutional neural network (CNN) models to detect dental decay based on X-ray images [5][6][7] or infrared transillumination images [8][9][10]. In recent years, several researchs have attempted to use color images of the oral cavity to automatically detect and classify dental decay based on AI [11][12]. In this research, we focus on detecting and classifying dental decay using deep learning models (experimental method), comparing the diagnostic performance of the models with expert assessments in color images.

This research paper focuses on analyzing and experimenting with three of the most widely used deep learning models for object detection today: Faster R-CNN, YOLO, and DETR. Specifically, we will introduce and analyze model information; present the results of these models with well-known available datasets such as PASCAL, COCO,... and our collected dataset.

2. **Some modern deep learning model.** eep Learning is a subset of Machine Learning that utilizes artificial neural networks with multiple layers (deep neural networks) to learn and automatically capture intricate features from data. Thanks to its ability to learn data representations automatically without human intervention, deep learning has achieved significant advancements, particularly in the field of object detection.

Before 2012, initial object detection methods focused on detecting objects using feature filtering techniques and traditional machine learning algorithms. Since 2012, with the emergence of AlexNet [13] and advanced techniques such as ReLU, Dropout, and LRN have achieved impressive results in object detection. From this point, object detection methods can be divided into two types: Single-stage object detectors, exemplified by the YOLO algorithms. [14] and multi-stage object detectors, which include several algorithms such as R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN,... [15]

By 2020, after making a significant impact in the field of Natural Language Processing (NLP), the Transformer was developed for application in object detection with the DETR model, opening up a new trend in object detection.

In this research paper, we will analyze three representative models for three trends in the object detection problem: Faster R-CNN for multi-stage object detection, YOLOv3 for single-stage object detection, and DETR for the end-to-end trend.

2.1. **Faster-RCNN.** Faster R-CNN [16] was introduced in 2016, developed by Shaoqing Ren and colleagues to address the speed and performance limitations of previous object detection models such as R-CNN and Fast R-CNN. Previously, R-CNN (2014) [17] was one of the first models to use CNN for object detection, but the process of processing each proposed region individually made the model very slow. Fast R-CNN (2015) [18] improved upon this by extracting features only once for the entire image; however, it still relied on manual region proposal methods like Selective Search, which limited its speed.

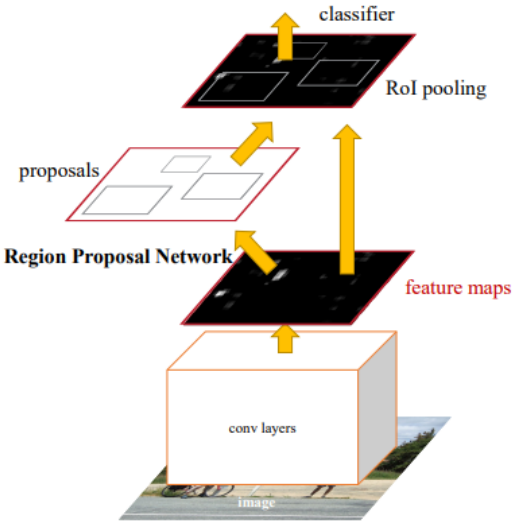

FIGURE 1. The Module RPN acts as the "attention"

Faster R-CNN is an advanced object detection model composed of two primary modules that work in tandem. The first module is a deep fully convolutional network known as the Region Proposal Network (RPN), which is responsible for generating region proposals. This network operates by scanning the entire input image, identifying and generating regions that are likely to contain objects. By applying convolutional layers, the RPN can highlight areas of interest within the image, thereby providing valuable information for rapid and accurate object detection.

The second module within the Faster R-CNN framework is Fast R-CNN, an improved method for object detection as introduced in a previous study [18]. The Fast R-CNN module processes the regions proposed by the RPN, classifying objects and predicting the bounding boxes for each detected object. Specifically, once it receives the proposed regions from the RPN, Fast R-CNN determines the type of object within each region and refines the positions to achieve more precise predictions. The structure of the Faster R-CNN model, illustrating how these two modules interact, is depicted in Figure 1

Region Proposal Networks (RPN). In Fast R-CNN, the region proposal process is carried out by traditional algorithms such as Selective Search or EdgeBoxes, which are costly and inefficient. The RPN addresses this issue by automatically generating region proposals during the training and prediction phases of the model.

The RPN takes an input image of any size and outputs a set of object boxes. The authors' idea is to share computation with the Fast R-CNN object detection module by using a common convolutional layer. To generate region proposals, we slide a small window, size 3 x 3 over the input feature map.

At each position of the sliding window, the model simultaneously predicts multiple region proposals. This approach assumes that the maximum number of proposals generated for each position is denoted as k. By predicting multiple proposals in one step, the model can efficiently identify potential object regions throughout the image, maximizing coverage across various object sizes and shapes.

Each region proposal is parameterized relative to a reference box known as an anchor. An anchor serves as a baseline or reference frame for proposals, allowing the network to estimate the bounding boxes relative to a predefined point in the image. Anchors are placed at the center of each sliding window position and are associated with specific aspect ratios and scales. This means that each anchor represents a potential object shape, enabling the model to predict objects of different sizes and orientations by adjusting these anchors.

For Fast R-CNN, the image is resized, and the feature map is computed for each scale, based on feature pyramids using HOG or deep convolutional networks (Figure 2). This method is usable but time-consuming in terms of computation. For Faster R-CNN, multi-

FIGURE 2. The pyramid of images and feature maps is constructed

scale sliding windows are used directly on the feature maps (Figure 3).

The bounding boxes are classified and regressed to different anchors. These boxes are based solely on the image and feature map with a single scale and use filters (sliding windows on the feature map) of a fixed size For each anchor, the RPN performs binary

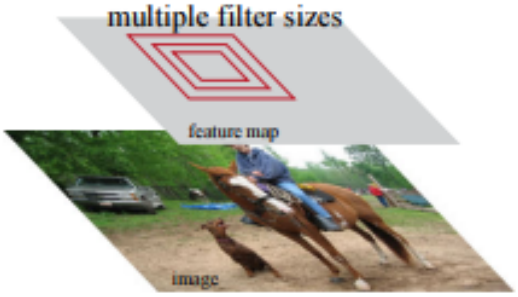

FIGURE 3. A pyramid of filters with multiple scales/sizes is applied to the feature map.

classification (object or no object). The loss function can be defined as follows:

$$L(\{pos_i\},\{truth_i\}) = \frac{1}{N_{\text{class}}}\sum_i L_{\text{class}}(pos_i, pos_i^*) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}(truth_i, truth_i^*). \quad (1)$$

Here, i represents the index of an anchor within the mini-batch, and pos_i denotes the predicted probability that anchor i contains an object. Label of ground-truth pos_i^* is 1 if anchor has object, is 0 if has no object. $truth_i$ i is the representing vector of four coordinate parameters of the predicted bounding box., $truth_i^*$ belongs to a ground-truth box associated with an anchor containing an object. Classification loss function L_{class} record the loss for two classes: with object and without object. With regression loss function, $L_{reg}(truth_i, truth_i^*) = R(truth_i - truth_i^*)$ where R is strong loss function(smooth L_1). $pos_i^* L_{reg}$ means that the regression loss function is activated for positive anchors ($pos_i^* = 1$) and is inactive in other cases ($pos_i^* = 0$). Output of cls class and reg class include $\{pos_i\}$ and $\{truth_i\}$ respectively.

L_{class} and L_{reg} is normalized respectively by N_{class} and N_{reg}, is weighted by a balancing parameter λ. class in (1) is normalized by mini-batch size (example: $N_{class} = 256$) and reg is normalized by number of anchor.

2.2. **YOLOv3.** YOLO [19] first introduced in 2015 by the author Joseph Redmon. One of his collaborators - Ross Girshick had introduced Faster R-CNN in previous time. They shared the ideas in computer vision research, therefore, there are some similarities with Faster R-CNN. However, while Faster R-CNN uses a two-stage detection model, the first version of YOLO does not have a region proposal step and is significantly faster than Faster R-CNN. Joseph Redmon developed the first three versions of YOLO. Due to personal reasons, this author was unable to develop any further versions of YOLO. Other authors have built upon Joseph Redmon's ideas to develop subsequent YOLO models. However, the architecture of these later YOLO versions has differed significantly from the original architecture proposed by Joseph Redmon. In this research, we will focus on analyzing the YOLOv3 model [20] - the final version of YOLO developed by Joseph Redmon—to investigate the ideas and original structure proposed by the author.

Bounding Box Prediction. YOLOv3 uses a technique called dimension clustering to optimize the prediction of bounding boxes, where anchor boxes serve as reference

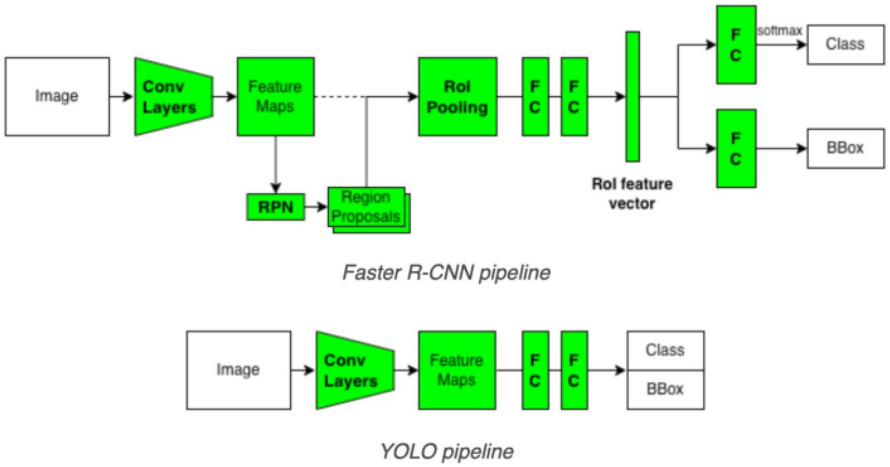

FIGURE 4. Overview of the Faster R-CNN and YOLOv1

templates. For each bounding box, the model predicts four coordinates: a_x, a_y, a_w, a_h. Specifically, if a grid cell is offset from the top-left corner of the image by (c_x, c_y) and the bounding box prior has a predefined width and height, denoted as p_w and p_h then the predictions are generated based on these reference points:

$$
\begin{aligned}
b_x &= \sigma(a_x) + c_x \\
b_y &= \sigma(a_y) + c_y \\
b_w &= p_w e^a_w \\
b_h &= p_h e^a_h
\end{aligned}
$$

YOLOv3 employs logistic regression to predict an objectness score for each bounding box, a method similar to that used in YOLOv2 [21]. This score represents the likelihood that a bounding box contains an object. The concept of predicting objectness scores in this manner draws inspiration from the Faster R-CNN model. The objectness score should ideally be 1 (indicating a positive label) if a bounding box prior has more overlap with a ground truth object than any other bounding box prior. This high score signals that the bounding box is well-aligned with an actual object in the image.

If a bounding box prior is not the optimal one but still has sufficient overlap with a ground truth object—exceeding a specified threshold—the prediction is ignored. YOLOv3 sets this threshold at 0.5. Additionally, YOLOv3 assigns only one bounding box prior to each ground truth object to avoid redundancy. If a bounding box prior is not matched to any ground truth object, the model does not incur a loss for the coordinate or class predictions for that box, and it only considers the objectness score, thus streamlining the prediction process.

Class Prediction. Each predicted bounding box assigns possible classes it may contain through multi-label classification. During training, the authors apply binary cross-entropy loss for label prediction. In this approach, softmax is avoided for label classification, as it assumes that each box can have only one label, which does not hold true for our dataset. This model extracts features from these scales using a method similar to Feature Pyramid Networks (FPN) [22]

Feature Extractor. YOLOv3 is used Darknet-53 network to feature extractor. Darknet-53 architecture is introduced in Figure 5

Predictions Across Scales. YOLOv3 predicts bounding boxes at three distinct scales of the feature map. The model utilizes a mechanism akin to Feature Pyramid Networks (FPN) to extract features from these scales. Specifically:

1. From the base feature extractor, several additional convolutional layers are added to refine the extracted features. The final convolutional layer produces a 3D tensor containing essential information about each bounding box, including coordinates, objectness scores, and class predictions.

2. Next, YOLOv3 upscales the feature map from the previous two layers by a factor of two, while also extracting a feature map from an earlier network layer. This map is then merged with the upscaled feature map through concatenation. The authors then apply several convolutional layers to process the concatenated feature map, resulting in a predicted tensor that mirrors the previous one, but with doubled dimensions.

3. This process is repeated again to predict bounding boxes at the final scale. As a result, predictions at the third scale leverage all prior computations along with detailed features from the earlier layers of the network.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

FIGURE 5. Overview of the Darknet-53 architecture

2.3. **DETR(DEtection TRansformer).** In 2017, the Transformer model [23] was introduced. It created a major breakthrough in the field of Natural Language Processing (NLP) due to its superiority in handling both text and time-series data. Leveraging the advantages of this model, in 2020, a research team at Facebook AI announced the DERT model. [24].

Upon its release, DETR was evaluated to have superior accuracy compared to Faster R-CNN and was fully competitive with recently launched YOLO models. Currently, the

authors have introduced several models that achieve high accuracy but are significantly more complex. This study will focus on analyzing the DETR model.

Set Prediction Loss for Object Detection. DETR produces a fixed number S of predictions in a single pass through its decoder, where S is chosen to be larger than the average object count per image. A main challenge during training is comparing these predictions (in terms of class, position, and size) to the ground truth. DETR's loss function solves this by first identifying an optimal one-to-one matching between predicted and ground truth objects, then applying object-specific loss terms, such as for bounding box losses.

Let h represent the set of ground-truth objects, and let $\hat{h} = \{\hat{h}_i\}_{i=1}^{S}$ denote a set of S predictions. Since S is chosen to be larger than the number of objects in the image, DETR treats h as a set of size S by padding it with empty (no-object) entries. To compute the correspondence between the sets h and $\hat{h}$, DETR searches for a permutation of S elements, $\sigma \in \sigma_N$, that minimizes the cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \sigma_S} \sum_{j}^{S} \mathcal{L}_{\text{match}}(h_j, \hat{h}_{\sigma(j)}) \tag{2}$$

Where $\mathcal{L}_{\text{match}}(h_j, \hat{h}_{\sigma(j)})$ is a pair-wise matching cost between ground truth h_i and a prediction with index $\sigma(j)$. This optimal assignment is computed efficiently with the Hungarian algorithm.

The next step is to calculate the loss function, specifically the Hungarian loss, for all pairs that were matched in the previous step:

$$\begin{aligned}\mathcal{L}_{\text{Hung}}(h, \hat{h}) = \sum_{i=1}^{N} \big[&- \log \hat{p}_{\hat{\sigma}(i)}(c_i) \\ &+ 1_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \big]\end{aligned} \tag{3}$$

where $\hat{\sigma}$ is the optimal assignment computed in the first step.

DETR architecture. The DETR architecture is notable for its simplicity, as shown in Figure 6. It consists of three key elements: a CNN backbone for extracting compact feature representations, an encoder-decoder transformer, and a simple feed-forward network (FPN) responsible for generating the final detection results.

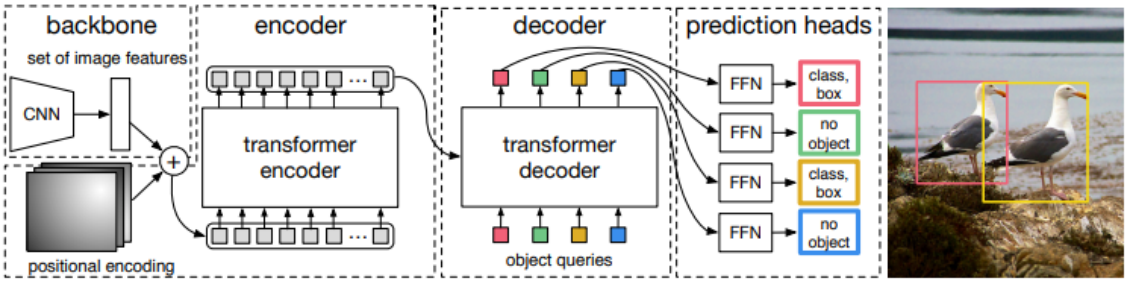

FIGURE 6. Overview DETR architecture

Backbone. Starting with the image $a_{img} \in R^{3 \times height_0 \times width_0}$ (having 3 channels), a standard backbone using CNN produces a lower-resolution activation map $f \in R^{C \times height \times width}$. Typical values for C are 2048, and height, width $= height_0/32, width_0/32$.

Transformer encoder. First, a 1x1 convolution reduces the channel dimension of the high-level activation map f from C to a smaller dimension d, resulting in a new feature map $z_0 \in R^{d \times height \times width}$. Since the encoder requires a sequence as input, the spatial

dimensions of z_0 are collapsed into a single dimension, yielding a $d \times height \times width$ feature map. Each encoder layer follows a standard architecture and consists of a multi-head attention module and a Feed Forward Network (FFN). Given that the transformer architecture is permutation-invariant, positional encodings need to be fixed and are added to the input of each attention layer.

Transformer decoder. In DETR, the decoder follows the typical Transformer architecture, processing S embeddings of dimension d using a combination of multi-head self-attention and encoder-decoder attention. Unlike the original Transformer, which generates the output sequence element by element in an autoregressive fashion, DETR processes all S objects at once in each layer of the decoder. To ensure unique outputs, the input embeddings must be distinct because of the permutation-invariant property of the decoder. These input embeddings, referred to as object queries, are learned positional encodings, and like in the encoder, they are added to the input at each attention layer. The decoder then transforms these S object queries into output embeddings, which are individually decoded into bounding box coordinates and class labels through a feed-forward network, resulting in S final predictions.

Prediction feed-forward networks - FFNs. The final prediction is generated by a 3-layer perceptron that uses ReLU activation and has a hidden dimension of d, followed by a linear projection layer. The Feed-Forward Network (FFN) [25] is responsible for predicting the normalized center coordinates, height, and width of the bounding boxes within the input image, while a linear layer applies a softmax function to assign class labels. Since a fixed number of bounding boxes, denoted as S, is predicted—typically much larger than the actual number of objects in the image—a special class, called the empty label, is added to account for slots where no object is present. This class serves a similar role to the background class in traditional object detection systems.

3. RESULTS.

3.1. **Dataset.** Panoramic RGB images were collected from patients aged 12 and older using advanced dental imaging equipment. To protect patient privacy and ensure security, the images were randomly chosen from the hospital's database, with no personal information taken into account. The images are annotated with three predefined labels:

FIGURE 7. Some sample images of dental cavities

Mild Decay, Moderate Decay and Severe Decay. The evaluation and labeling processes are guided and supervised by experts in the field of dentistry.

Our dataset contains a total of 1,980 color images divided into two parts as follows:
(1) Training dataset: 1387 images
(2) Testing dataset: 503 images

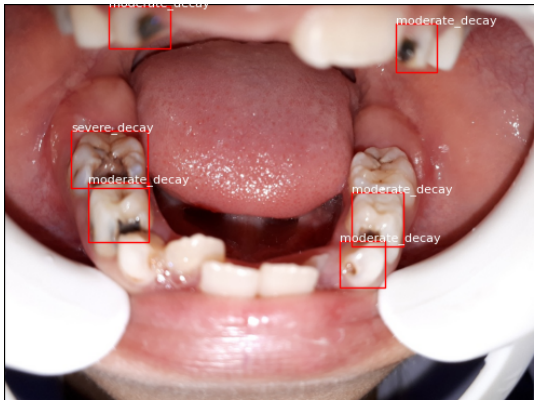

FIGURE 8. Labeled images of dental cavities

Due to the unique characteristics of the dataset, which have not yet appeared in domestic or international research, we have tentatively named this dataset "DECAY PTIT 2024" (DP2024).

3.2. **Experimental Setup and Configuration.** The deep learning models analyzed above all use a CNN to extract image feature information. To increase objectivity in model evaluation, this study uses different feature extraction networks for each model. However, for YOLOv3, which has limited flexibility in feature extraction networks, we retain the original Darknet-53 network. The CNN networks used in this research are: VGG16 [26], Resnet50, Resnet101 vand Resnet152 [27]. These are classic models that have been considered effective for image recognition in recent years.

The models were trained with a comparable number of epochs to facilitate the most intuitive comparison of their effectiveness. In the original studies, the authors kept the learning rate hyperparameter constant across the models. This led to the situation where, after a certain time limit, the model would no longer converge, requiring a smaller learning rate.

There are several methods to adjust the learning rate. Typically, the Adam optimizer function [28] (Equation (4)) is used to automatically support the adjustment of the learning rate.

$$\begin{aligned} g_t &= \Delta_\theta \mathcal{L} \\ w_t &= \gamma_1 w_{t-1} + (1-\gamma_1) g_t \\ n_t &= \gamma_2 n_{t-1} + (1-\gamma_2) g_t^2 \\ \theta_t &= \theta_{t-1} + \frac{\eta}{\sqrt{\frac{n_t}{1-\beta_2^t} + \epsilon}} \times \frac{w_t}{1-\beta_1^t} \end{aligned} \tag{4}$$

However, using Adam requires a small learning rate, which leads to slow convergence. Additionally, Adam has poor generalization with complex data, making it unsuitable for our dataset. Besides Adam, another commonly used normalization function is SGD (Stochastic Gradient Descent) combined with momentum. The overall formula for SGD is expressed as follows:

TABLE 1. Results of the models with different backbones.

Model	Backbone	Is dynamic lr	COCO-mAP	DP2024-mAP	DP2024-Loss
Faster RCNN	VGG16	No	.42	.45	.26
Faster RCNN	VGG16	Yes	-	.50	.24
Faster RCNN	Resnet101	No	.36	.11	.32
Faster RCNN	Resnet101	Yes	-	.14	.32
YOLOv3	Darknet-53	No	.33	.28	.45
YOLOv3	Darknet-53	Yes	-	.31	.41
DETR	Resnet50	No	.43	.07	2.15
DETR	Resnet50	Yes	-	.08	2.44
DETR	Resnet152	No	.45	.08	2.51
DETR	Resnet152	Yes	-	.10	2.47

$$
\begin{aligned}
g_{normalized} &= \frac{g}{||g||_p} \\
v &= \beta v + (1-\beta) g_{normalized} \\
\theta_t &= \theta_{t-1} - \eta v
\end{aligned} \tag{5}
$$

SGD is relatively simple, easy to understand, converges quickly, and generalizes well to data. However, it still has a significant drawback: a fixed learning rate. If the learning rate is initialized too high, it will prevent the algorithm from converging, causing it to oscillate around the target due to the large step size; conversely, a small learning rate will negatively affect the training speed.

This research proposes adjusting the learning rate during the training process using the SGD optimizer combined with momentum. The method we introduce has two main phases.

First phase: First, we proceed to initialize the initial learning rate (lr_{init}) with a value of 1e-2 for each model. During this phase, the learning rate will gradually increase from a very small value to lr_{init} in some of the initial batches, determined by the parameter *burn_in*

$$
\begin{aligned}
lr &= lr_{init} \times \frac{batches\_done}{burn\_in} \\
&(batches\_done < burn\_in)
\end{aligned} \tag{6}
$$

Where:

- lr_{init}: initial learning rate value
- *batches_done*: total number of batches processed up to the current batch
- *burn_in*: number of batches in the burn-in phase

Second phase: decrease gradually according to learning rate milestones (lr_{steps}) and update the learning rate. After *burn_in* phase, if *batches_done* exceeds certain milestones specified in lr_{steps}, the learning rate will gradually decrease at each milestone.

$$
lr = lr_{init} \times \prod_{\text{threshold} \leq \text{batches\_done}} \text{value} \tag{7}
$$

With each threshold inWith each threshold in lr_{steps}, if *batches_done* exceeds that threshold, the learning rate will be multiplied by the corresponding value. After calculating the new learning rate, this value will be reassigned to each parameter group in the optimizer.

This study uses Mean Average Precision (mAP) and the loss on the testing dataset as evaluation metrics due to their relevance for assessing our dataset.

3.3. **Comparison of Results Among Models.** The research sequentially compares the results based on the following situations:

- Evaluation and detailed analysis of the model based on mean average precision and other metrics.
- Evaluate the size and processing speed of the model.

1) Evaluation and detailed analysis of the model based on mean average precision and other metrics

To assess the effectiveness of the proposed improvements and to evaluate how well the model adapts to the DP2024 dataset, the research involves a comprehensive comparison of key metrics. Specifically, we compare the performance of a model trained with a fixed learning rate to that of a model trained with a variable learning rate, as proposed in our approach. This comparison aims to determine whether the adaptive learning rate improves model convergence and overall performance on the DP2024 dataset.

Furthermore, to gain a broader perspective on the model's capabilities, we also benchmark its performance on the prepared DP2024 dataset against the widely used COCO dataset [29]. The COCO dataset, being a well-established benchmark in the field, provides a valuable comparison point, enabling us to understand how the model performs relative to other state-of-the-art models that have been trained and evaluated on this dataset.

The Mean Average Precision (mAP) and loss metrics are then calculated. Since this measurement takes into account the ability to match randomly and the correlation of the degree of deviation, we believe that it is more reliable and convincing than pure accuracy.

To confirm the effectiveness of the proposed method, we conducted a comparison based on the metrics presented in Table I. It can be observed that the proposed method, which involves changing the learning rate during the training process, is more effective than using a fixed learning rate. Specifically, the mAP scores of all three models show improvements ranging from 1% to 5%.

By comparing results on the DP2024 dataset and the COCO dataset, we observe distinct differences across various models. For Faster R-CNN, the model demonstrates outstanding performance on our dataset, achieving an mAP score higher by 5% and 3% respectively on the VGG16 and ResNet101 backbones. This indicates that the Faster R-CNN model is well-suited for our dataset, which features small object regions with minimal differences in the surrounding areas.

For the YOLOv3 model, experiments show that the results on the DP2024 dataset are not significantly different from those on the COCO dataset. Due to YOLOv3's "one-stage" approach, this model does not have the process of generating object proposals like Faster R-CNN, leading to missed detections or less accurate identification of complex objects. Given the characteristics of our dataset, which has a high degree of diversity and where the size of the objects is much smaller compared to the image dimensions, YOLOv3 yields a relatively low mAP score along with a considerable loss.

Regarding DETR, despite the significant improvements in metrics when changing the backbone or the learning rate, this model does not perform well on the DP2024 dataset. The main reason is that DETR requires a very large dataset (over 300,000 images) like COCO, whereas our dataset contains only about 2,000 images, failing to meet the model's

requirements. Additionally, DETR also necessitates a long training time with a large number of epochs (over 500 epochs). Due to limitations in time and resources, this study has not been able to meet the demands of the DETR model, resulting in unsatisfactory outcomes when implemented in practice.

2) Evaluate the size and processing speed of the model

The predicted results for dental decay images obtained from this experimental study are presented in Figures 9, 10, and 11, showing the outcomes sequentially for the Faster R-CNN, YOLOv3, and DETR models.

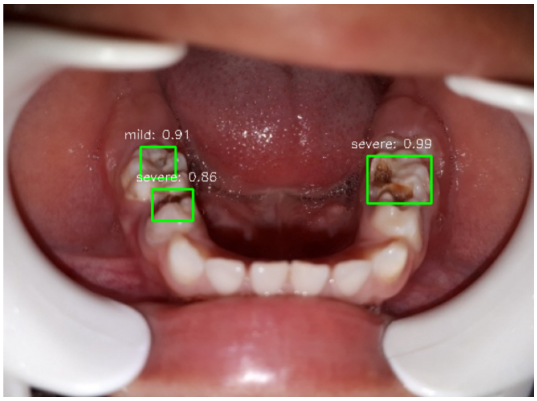

FIGURE 9. Result of Faster R-CNN model

As seen in Figure 9, the Faster R-CNN model with the proposed improved method achieved the best accuracy of 0.99 on dental decay images with small sizes. Additionally, the models also had a prediction processing speed of 0.1 seconds per image.

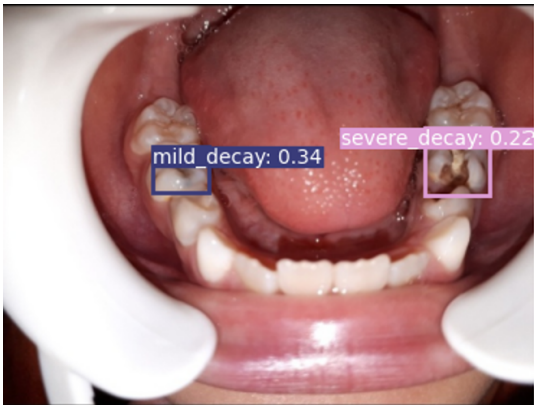

FIGURE 10. Result of YOLOv3 model

The results of the YOLOv3 model are shown in Figure 10. The model clearly distinguished between the different levels of dental decay; however, its confidence scores were not very high (ranging from 22% to 34%), indicating that these results may not be entirely accurate and require further manual verification by a dentist.

Figure 11 shows the results from the DETR model. It can be seen that the bounding boxes are trending towards marking the locations of dental decay, but they are still significantly off and the accuracy is very low. To improve the accuracy of the DETR model, a substantial increase in the dataset size is needed, which requires considerable time and effort. Therefore, using DETR for detecting dental decay based on the DP2024 dataset is not practical.

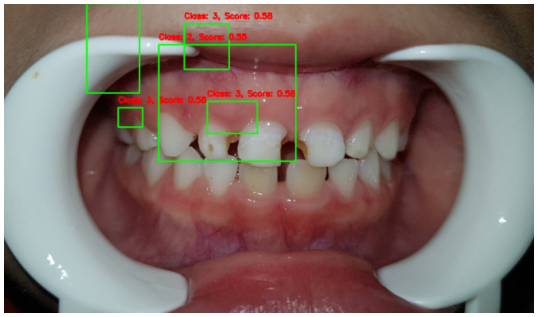

FIGURE 11. Result of DETR model

Based on these results, the use of modern models with a method of varying learning rates during the training process for the object detection task with a small dataset has proven to be feasible and yielded promising results. This approach can be applied in practice, providing a relatively high accuracy method for assisting in the detection of dental decay in children.

4. **CONCLUSION.** This research has applied advanced deep learning models such as Faster R-CNN, YOLOv3, and DETR to detect dental caries in children. The results from the comparative table indicate that models with different backbones have varying levels of effectiveness. Notably, the use of a dynamic learning rate strategy significantly improved the mAP accuracy on the DP2024 dataset for both the Faster R-CNN and YOLOv3 models, as clearly reflected in the mAP and loss values. Specifically, Faster R-CNN with VGG16 and ResNet101 backbones showed improvements in mAP and reduced loss when employing the dynamic learning rate compared to not using it. For YOLOv3, the improvement in mAP, although small, is still significant with the use of a dynamic learning rate. Regarding the DETR model, while the mAP values on the DP2024 dataset are still low, models with the ResNet152 backbone demonstrate potential with a slight improvement in accuracy compared to ResNet50.

REFERENCES

[1] P. Salagare and et al, "An overview of internet of dental things: New frontier in advanced dentistry," *Wireless Pers Commun*, vol. 110, pp. 1345–1371, 2020.

[2] M. Auderset, Connert and et al, "Evaluation of five methods to identify composite restorations in human teeth on a forensic purpose—an ex vivo comparative study." *Int J Legal Med*, vol. 138, pp. 85–96, 2024.

[3] Kuhnisch J, Meyer O, Hesenius M, Hickel R, and Gruhn V, "Caries detection on intraoral images using artificial intelligence," *Journal of Dental Research*, vol. 101, pp. 158–165, 2022.

[4] Hickel, R., Mesinger, S., Opdam, and N. et al, "Revised fdi criteria for evaluating direct and indirect dental restorations—recommendations for its clinical use, interpretation, and reporting." *Clin Oral Invest*, vol. 27, pp. 2573–2592, 2024.

[5] A. Imak, A. Celebi, K. Siddique, M. Turkoglu, A. Sengur, and I. Salam, "Dental caries detection using score-based multi-input deep convolutional neural network," *IEEE*, vol. 10, pp. 18 320–18 329, 2022.

[6] Forouzeshfar, Safaei, and Ghaderi et al, "Dental caries diagnosis using neural networks and deep learning: a systematic review," *Multimed Tools Appl*, vol. 83, pp. 30 423–30 466, 2024.

[7] Zhu, Cao, and Lian et al, "Cariesnet: a deep learning approach for segmentation of multi-stage caries lesion from oral panoramic x-ray image." *Neural Comput Applic*, vol. 35, pp. 16 051–16 059, 2023.

[8] Haghanifar, Majdabadi, and Haghanifar et al, "Paxnet: Tooth segmentation and dental caries detection in panoramic x-ray using ensemble transfer learning and capsule classifier," *Multimed Tools Appl*, vol. 82, pp. 27 659–27 679, 2023.

[9] Dewangan, D.K., Sahu, and S.P, "Rcnet: road classification convolutional neural networks for intelligent vehicle system," *Intel Serv Robotics*, vol. 14, pp. 199–214, 2021.

[10] S. Ho et al, "Dlam: Deep learning based realtime porosity prediction for additive manufacturing using thermal images of the melt pool," *IEEE*, vol. 9, pp. 115 100–115 114, 2021.
[11] Cui, Z., Fang, Y., May, and L. et al, "A fully automatic ai system for tooth and alveolar bone segmentation from cone-beam ct images," *Nat Commun*, vol. 13, p. 2096, 2022.
[12] Park, E.Y., Jeong, S., Kang, and S. et al, "Tooth caries classification with quantitative light-induced fluorescence (qlf) images using convolutional neural network for permanent teeth in vivo," *Nat Commun*, vol. 23, p. 981, 2023.
[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012.
[14] Ghazlane Yasmine, Gmira Maha, and Medromi Hicham, "Overview of single-stage object detection models: from Yolov1 to Yolov7," *IEEE*, 2023.
[15] Lixuan Du, Rongyu Zhang, and Xiaotian Wang, "Overview of two-stage object detection algorithms," *Journal of Physics Conference Series*, 2020.
[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.
[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *IEEE Computer Vision and Pattern Recognition*, 2014.
[18] Ross Girshick, "Fast R-CNN," *IEEE International Conference on Computer Vision*, 2015.
[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.
[20] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," *IEEE International Conference on Computer Vision*, 2018.
[21] Ali Farhadi and Joseph Redmon, "YOLO9000: Better, Faster, Stronger," *IEEE International Conference on Computer Vision*, 2016.
[22] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature Pyramid Networks for Object Detection," *IEEE International Conference on Computer Vision*, 2017.
[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need," 2017.
[24] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko, "End-to-End Object Detection with Transformers," 2017.
[25] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le, "Attention Augmented Convolutional Networks," *IEEE International Conference on Computer Vision*, 2019.
[26] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *IEEE Computer Vision and Pattern Recognition*, 2014.
[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," *IEEE Computer Vision and Pattern Recognition*, 2016.
[28] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *International Conference for Learning Representations*, 2014.
[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár, "Microsoft COCO: Common Objects in Context," *European Conference on Computer Vision*, 2014.