Journal of Information Hiding and Multimedia Signal Processing ©2025 ISSN 2073-4212
Ubiquitous International Volume **16**, Number **1**, March **2025**

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

Received November 2, 2024, revised January 16, 2025, accepted January 20, 2025.

Abstract. *Dental decay is one of the most common oral health problems worldwide, but datasets about it are very limited. This paper introduced a new dataset for dental decay, which included images of children over 12 years old from many dental facilities in Viet Nam. At the same time, we also introduced a new change learning rate method for SGD optimizer. To verify the effectiveness of the proposed method, we compare the results with the Adam optimizer to evaluate the differences in convergence speed and accuracy based on training 3 models: Faster R-CNN, YOLOv3 and DETR, without using pretrained weights*

Keywords: Detect decay, Health care, Deep learning, Object detection

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, YOLOv3, and DETR. Specifically, we will introduce and analyze model information; present the results of these models with well-known available dataset is COCO,... and our collected dataset.

2. RELATED WORK.

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 trending.

2.1. Faster-RCNN.

Faster R-CNN [16] is an advanced object detection model composed of two primary modules. 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, there by providing valuable information for rapid and accurate object detection.

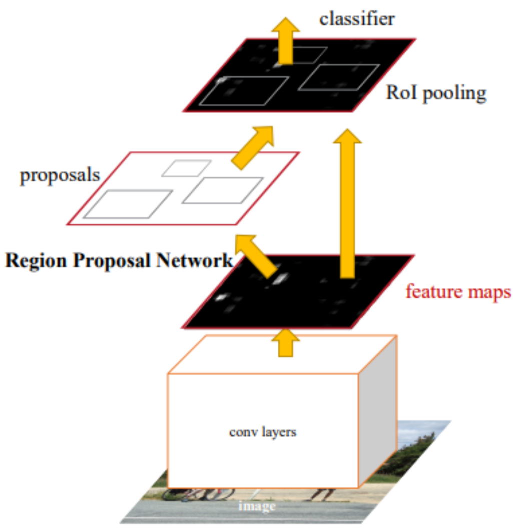

FIGURE 1. The Module RPN acts as the "attention"

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 [17]. 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. 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-

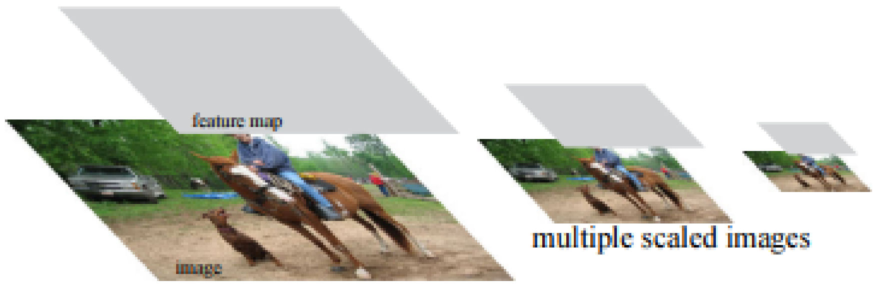

FIGURE 2. The pyramid of images and feature maps is constructed

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

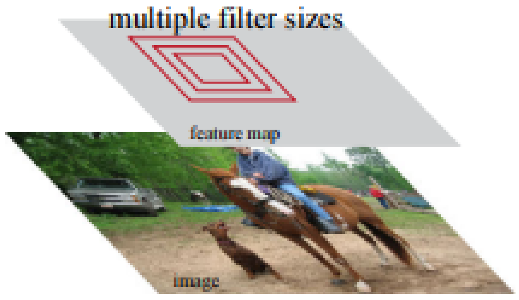

FIGURE 3. A pyramid of filters with multiple scales/sizes is applied to the feature map.

Loss Function for RPN. 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.** YOLOv3 [18] shared the ideas with Faster R-CNN is using anchor boxes. However, while Faster R-CNN uses a two-stage detection model, YOLOv3 does not have a region proposal step and is significantly faster than Faster R-CNN.

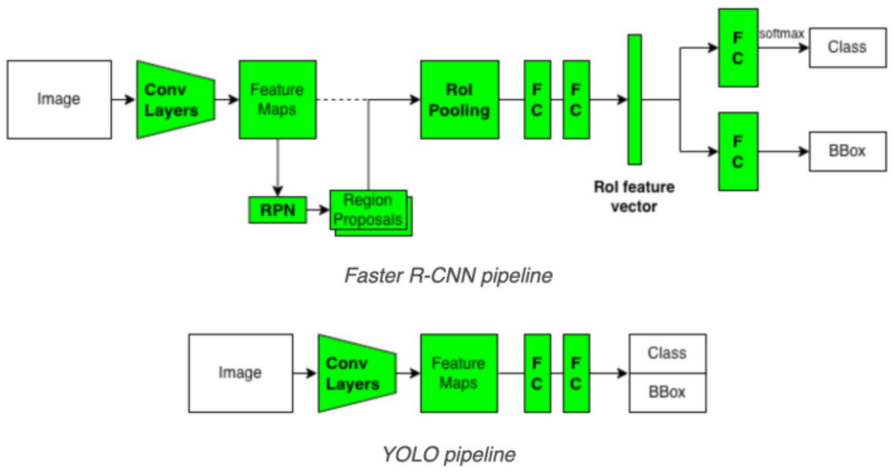

FIGURE 4. Overview of the Faster R-CNN and YOLO pipeline

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

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

YOLOv3 employs logistic regression to predict an objectness score for each bounding box, a method similar to that used in YOLOv2 [19]. 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) [20]

Feature Extractor. YOLOv3 is used Darknet-53 network to feature extractor.

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.

2.3. DETR(DEtection TRansformer).

Leveraging the advantages of Transformer model [21], in 2020, a research team at Facebook AI announced the DERT model. [22].

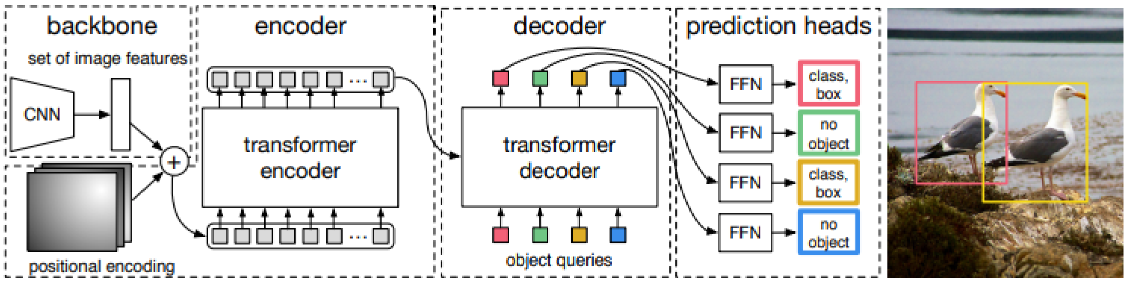

FIGURE 5. Overview DETR architecture

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.

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. The DETR decoder uses a Transformer architecture to process S embeddings simultaneously rather than sequentially. Unique predictions are ensured through distinct learned positional encodings called object queries, added at each attention layer. The decoder converts these queries into output embeddings, which are decoded into bounding box coordinates and class labels via a feed-forward network, yielding S predictions.

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:

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

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

3. DATASET AND CHANGING LEARNING RATE METHOD DESCRIPTIONS.

3.1. Dataset.

Panoramic RGB images were taken from patients 12 years old and older using advanced dental imaging equipment. To protect patient privacy and ensure security, images were randomly chosen from the hospital's database, without personal information being taken into account.

The images are annotated with three predefined labels: enamel decay, dentin decay, and pulpitis decay. The evaluation and labeling processes are guided and supervised by experts in the field of dentistry. We can see the description of each label in Figure 6.

Signs of enamel decay are the appearance of opaque white streaks, black spots, or brown spots on the surface of the tooth, which are often easily visible to the naked eye (Figure 6-A). With dentin decay, the typical symptom is the appearance of cavities in the tooth, leading to loss a part of the tooth (Figure 6-B). In the case of pulpitis decay, In the case of deep dentin caries, deep black cavities can clearly be seen penetrating into the tooth root, causing the tooth to lose its original shape.

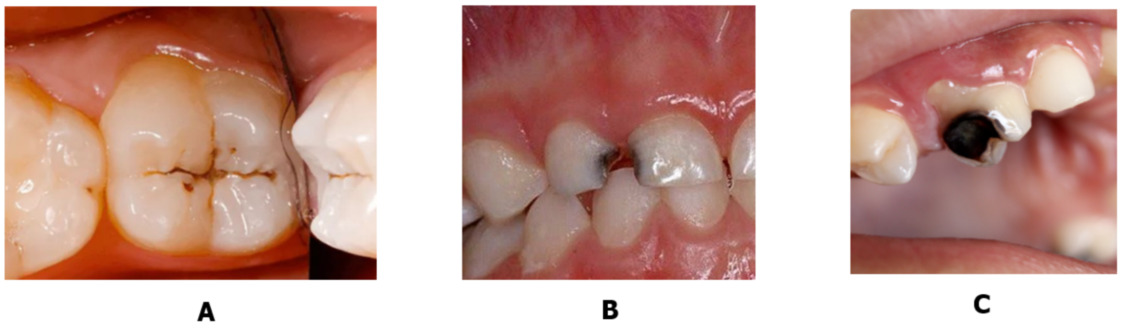

FIGURE 6. Example for each label. Enamel decay (6-A), Dentin decay (6-B), Pulpitis decay (6-C)

Our dataset contains a total of 1,981 color images divided into two parts as follows:

(1) Training dataset: 1387 images

(2) Testing dataset: 504 images

Due to the unique characteristics of the dataset, which have not yet appeared in domestic or international research, we have tentatively named this dataset: Tooth Decay PTIT 2024 (TDP2024). Figure 7 was showed the distribution of images for each label in our dataset

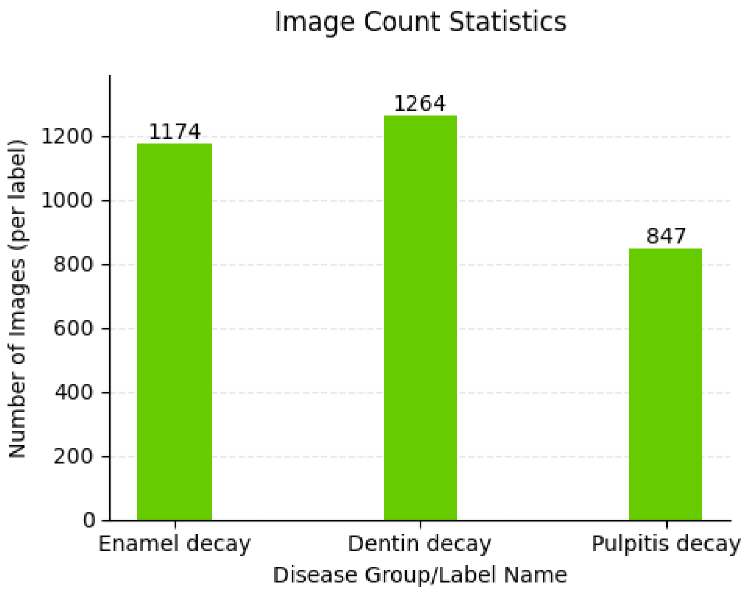

FIGURE 7. Statistics of the number of images for each label

3.2. **Changing learning rate method.** There are several methods to adjust the learning rate. Typically, the Adam optimizer function [23] (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:

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

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.

Our research introduced a new menthod for adjusting the learning rate during the training process use SGD optimizer combined with momentum. Current optimization methods typically assume that the derivative of the function in the model is at a point approaching a maximum or on a flat region of the function. When updating parameters, the derivative point gradually moves toward the minimum. However, this assumption

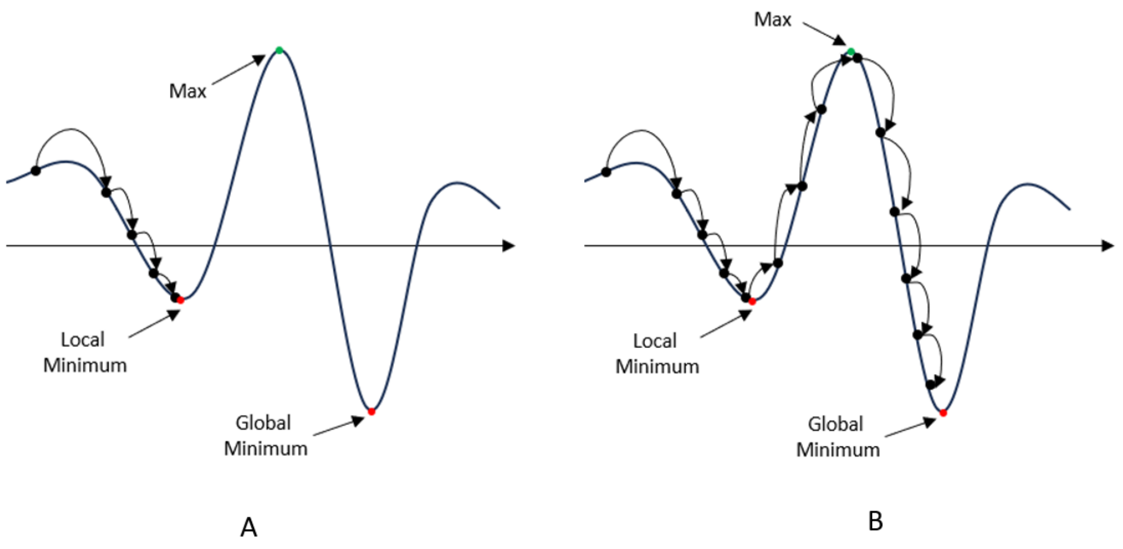

FIGURE 8. Assumption of another methods (8-A) and our method motivation (8-B)

fails to address the practical issue where some derivative points reach local minima but cannot overcome steep slopes of the function to progress toward the global minimum.

The idea of our proposed method is based on the assumption that the derivative is at a local minimum and needs to be elevated to a higher value to overcome the adjacent maximum. To avoid the case where the derivative fails to converge, we also implement an adaptive learning rate after pushing this parameter past the maximum of the derivative. The motivation of this method can be compare with another methods in Figure 8. Figure 8-A illustrates for assumption of another optimizer method like: SGD, Adam,... Figure 8-B describe our method.

Our method has two 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*. Target of this phase is help derivative point can get out local minimum and over the max point of function.

$$lr = lr_{init} \times \frac{batches\_done}{burn\_in} \qquad (6)$$
$$(batches\_done < burn\_in)$$

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.

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

With each threshold in 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.

3.3. **Dental caries detection overview.** Although there are differences in their initial structures, when applied to the problem of dental caries detection, all these methods share a basic architecture as described in Figure 9

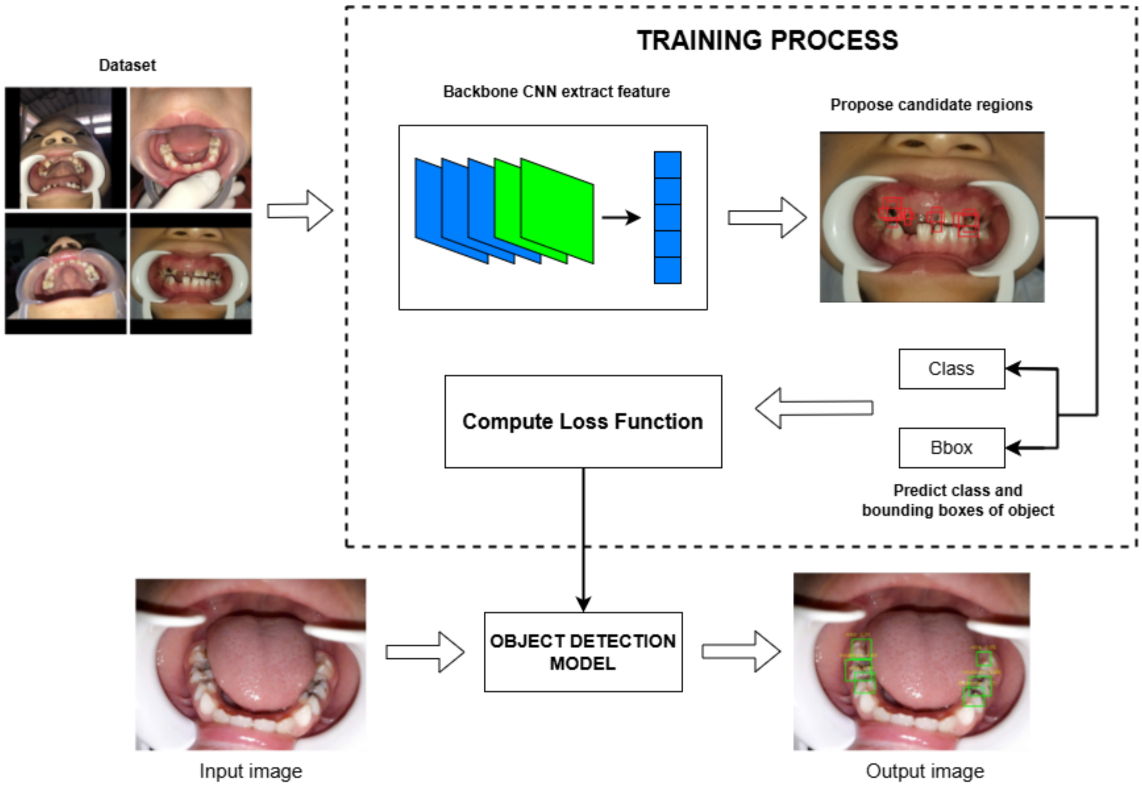

FIGURE 9. Overview of Dental caries detection

First, a set of images is fed into the training process. During training process, our research using batch size is 16. This means that each data batch fed into one epoch consists of 16 images. This does not imply that 16 images are the optimal choice, but it is due to the limitations of our hardware.

These images are passed through a convolutional neural network to extract important features, resulting in feature maps. After feature extraction, the model proposes candidate regions containing potential objects. Faster R-CNN and YOLOv3 utilize anchor boxes, while DETR uses object queries to detect objects.

During training, the model is optimized based on a loss function. This loss function guides the training process by adjusting the model's parameters to minimize the error between predictions and ground truth labels.

At the end of the training process, a model is obtained for object detection. The next section will present the training process and the results of the models.

4. RESULTS.

4.1. **Experimental setup and Training process.** The deep learning models analyzed above all use a CNN to extract information from image feature. 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 [24], Resnet50, Resnet101 vand Resnet152 [25]. These are classic models that have been considered effective for image recognition in recent years.

This study uses mean Average Precision (mAP) and Loss in the test dataset as evaluation metrics due to their relevance for assessing our dataset.

Figure 10 and Figure 11 are overviews for the training process using Adam optimizer and our method.

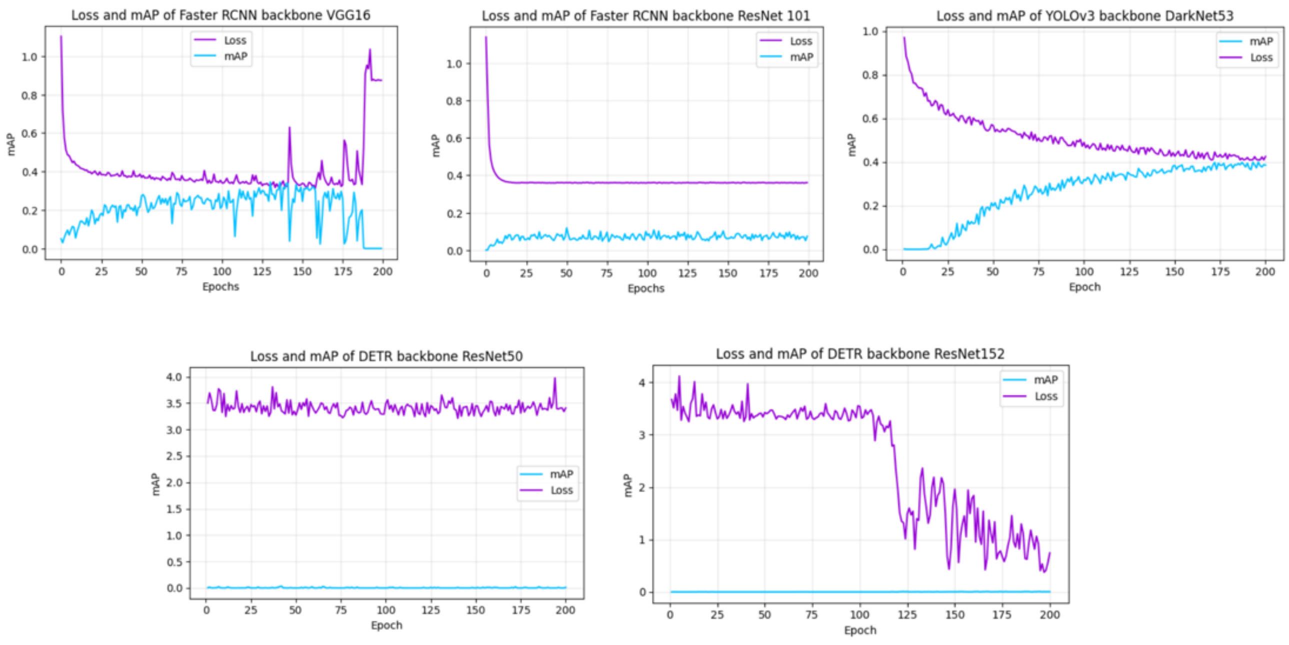

FIGURE 10. Training process with models and multi backbones using Adam optimizer

It is evident that during the training process of the Faster R-CNN model with both VGG16 and ResNet101 backbones, the Adam optimizer results in relatively low mAP and high Loss values. Notably, when using VGG16, in the final epochs of training, the Loss increases significantly, and the mAP drops to approximately 0, indicating an abnormality in the training process. With ResNet101 as the backbone, the model's mAP remains low, and from around the 20th epoch onward, both Loss and mAP show almost no change, suggesting signs of overfitting.

For YOLOv3, the Adam optimizer provides relatively stable training results, with Loss and mAP still showing a tendency to improve with continued training. However, for the DETR model, the Loss is extremely high, and the mAP is very low, indicating ineffective learning. Furthermore, with the ResNet50 backbone, the Loss generally does not decrease during training, showing that the model fails to learn. On the other hand, with the ResNet152 backbone, while the Loss decreases, it remains significantly high relative to the mAP.

After applying the proposed learning rate adjustment method to the models, the training results were very promising, especially with the Faster R-CNN model. With the VGG16 backbone, the model achieved significantly higher metrics compared to Adam, and the convergence speed was faster, requiring just over 30 epochs before overfitting began.

For YOLOv3, the model converged faster than when using Adam; however, the mAP score was lower.

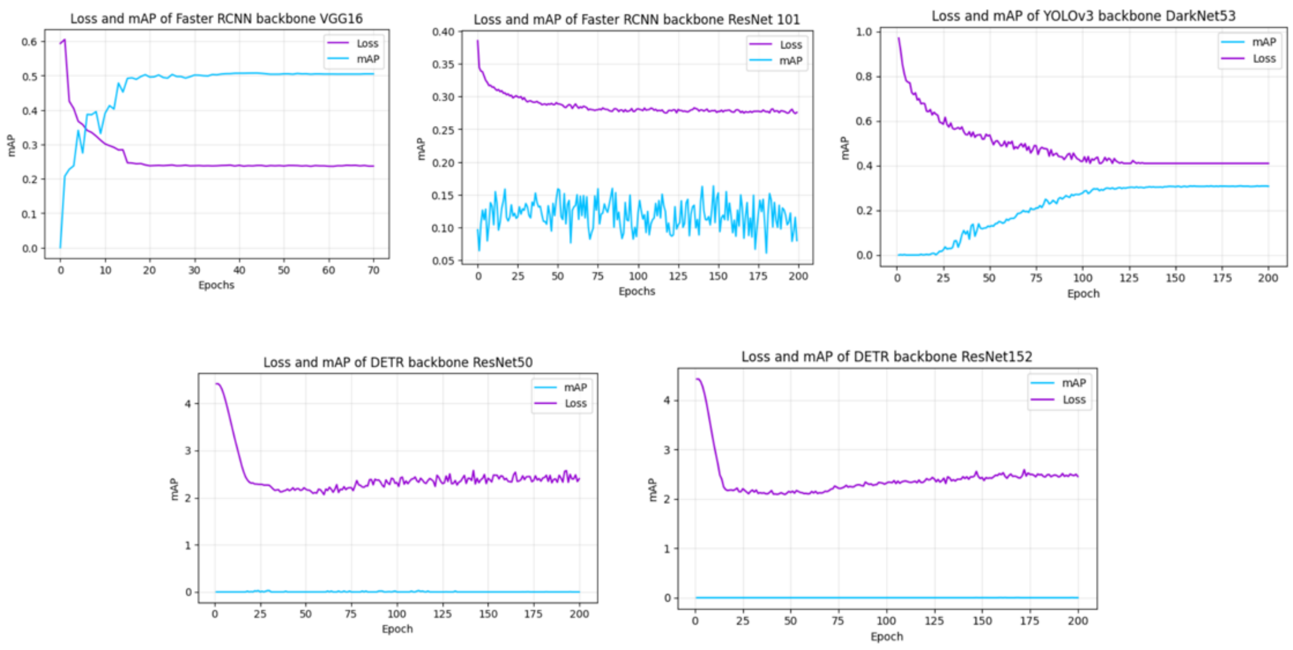

FIGURE 11. Training process with models and multi backbones using our method

For the DETR model, the results of applying the proposed method were not significant. The Loss decreased during training, indicating that the model was able to learn, but it remained very high. While the mAP score improved, the overall effectiveness of the model was still poor.

4.2. **Comparison of Results Among Models.** To assess the effectiveness of the proposed improvements and to evaluate how well the model adapts to the TDP2024 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 TDP2024 dataset against the widely used COCO dataset [26]. 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 three models show improvements ranging from 1% to 5%.

By comparing results in the DP2024 dataset and the COCO dataset, we observe distinct differences across various models. For Faster R-CNN, the model shows outstanding performance in our dataset, achieving an mAP score higher than 5% and 4%, respectively on the VGG16 and ResNet101 backbones. This indicates that the Faster R-CNN model

Table 1. Results of the models with different backbones.

Model	Backbone	Is dynamic lr	COCO-mAP	DP2024-mAP	DP2024-Loss
Faster RCNN	VGG16	No	0.42 [17]	0.38	0.33
Faster RCNN	VGG16	Yes	-	0.50	0.24
Faster RCNN	Resnet101	No	0.36 [18]	0.11	0.39
Faster RCNN	Resnet101	Yes	-	0.16	0.27
YOLOv3	Darknet-53	No	0.33 [18]	0.40	0.40
YOLOv3	Darknet-53	Yes	-	0.31	0.41
DETR	Resnet50	No	0.43 [22]	0.08	3.30
DETR	Resnet50	Yes	-	0.10	2.40
DETR	Resnet152	No	0.45 [22]	0.06	0.40
DETR	Resnet152	Yes	-	0.07	2.50

is well-suited for our dataset, which features small object regions with minimal differences in the surrounding areas.

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. Furthermore, DETR also requires a long training time with a large number of epochs (more than 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 results when implemented in practice.

Moreover, the results also indicate that the deeper the CNN-based backbone, the poorer the performance on this dataset. This can be explained as follows: A deeper network has a larger number of trainable parameters; with a small dataset, the training process is insufficient to update these parameters to match the distribution of the TDP2024 dataset, even when these backbones have been pre-trained on the ImageNet dataset. Therefore, in the results, the VGG16 backbone demonstrates an appropriate number of trainable parameters for this dataset.

The predicted results for dental decay images obtained from this experimental study are presented in Figures 12, showing the results sequentially for the Faster R-CNN, YOLOv3, and DETR models.

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

The results of the YOLOv3 model are clearly distinguished between the different levels of dental decay; however, its confidence scores were not very high (ranging from 22% to

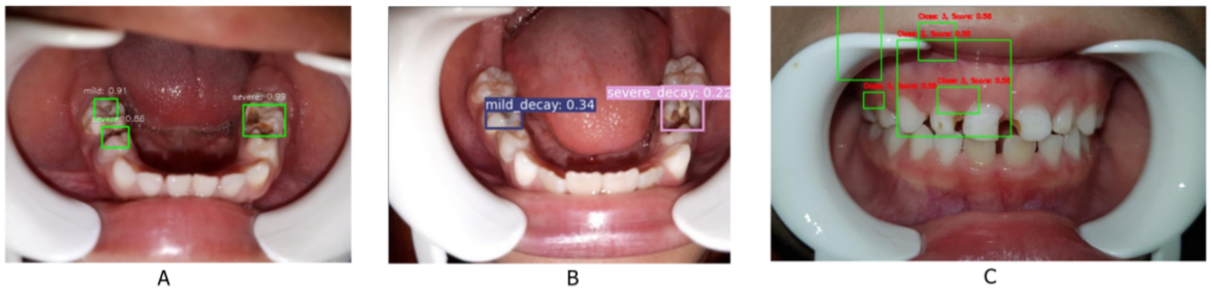

FIGURE 12. Results of models: Faster R-CNN (11-A), YOLOv3 (11-B) and DETR (11-C)

34%), indicating that these results may not be completely accurate and require further manual verification by a dentist.

DETR model's result 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 size of the dataset is needed, which requires considerable time and effort. Therefore, using DETR to detect dental decay based on DP2024 is not practical.

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 feasible and yielded promising results. This approach can be applied in practice, providing a relatively high-accuracy method to assist in the detection of dental decay in children.

5. **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 of the comparative table indicate that models with different backbones have varying levels of effectiveness. In particular, the use of a dynamic learning rate strategy significantly improved the mAP accuracy in 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. This research also has some limitations as follows:

- This research was conducted on a dataset with a very small number of labels (3 labels: Mild decay, Moderate decay, and Severe decay). This limits the study's applicability to a broader range of pathologies, confining it only to the conditions present in the dataset.
- The proposed learning rate adjustment method in the study, while capable of improving the accuracy of some models, was not adaptable to certain others (e.g. the YOLOv3 model in the study). This indicates that the proposed method has potential for further development but lacks a solid mathematical foundation to ensure consistent improvement.

- Although the learning rate adjustment method improved accuracy, it did not address the issue of small datasets in some related models, such as Transformer-based architectures. For example, in the DETR model, while accuracy improved, the results were still insignificant for commercial application.
- Techniques to overcome accuracy limitations, such as few-shot learning and transfer learning, have not yet been applied to enhance the models' performance.

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, "Fast R-CNN," *IEEE International Conference on Computer Vision*, 2015.

[18] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," *IEEE International Conference on Computer Vision*, 2018.

[19] Ali Farhadi and Joseph Redmon, "YOLO9000: Better, Faster, Stronger," *IEEE International Conference on Computer Vision*, 2016.

[20] 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.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need," 2017.

[22] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko, "End-to-End Object Detection with Transformers," 2017.

[23] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *International Conference for Learning Representations*, 2014.

[24] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *IEEE Computer Vision and Pattern Recognition*, 2014.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," *IEEE Computer Vision and Pattern Recognition*, 2016.

[26] 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.