Course: Digital Image Processing Applications

Student: Allahverdi Hajiyev (98716610993)

Project: Dental Disease Detection

**ABSTRACT**

The paper discusses the technological approach to dental disease diagnosis. Considering the fact that many dental diseases have been emerged over the last decades, it becomes really difficult to identify them correctly in real life. The goal of the project is to detect dental disease from a panoramic view of the mouth which will lead to accurate diagnosis. As a result of this project, doctors will be able to understand the case more clearly and prevent any progress of the disease. Doctors are also humans so they can make mistakes during the diagnosis. However, using the technological advancement, we can build AI models which will reduce the number of such cases. As a result, the project will prevent doctors from making wrong decisions about the illness since they will have more accurate model to decide instead of them. On the other hand, the project might be used by doctors to double check their diagnosis.

**CONTENT**

**INTRODUCTION5**

Executive Summary5

Related works5

Object Detection Algorithms13

R-CNN13

Mask R-CNN14

YOLO15

YOLOv116

YOLOv218

YOLOv319

YOLOv421

YOLOv522

YOLOv624

YOLOv725

Detectron225

**DEVELOPMENT29**

**RESULTS & DISCUSSION32**

**CONCLUSION35**

**REFERENCES36**

**LIST OF ABBREVIATIONS AND CONVENTIONAL SIGNS40**

**INTRODUCTION**

**Executive Summary**

In the recent world, people often face problems in medicine which are related to wrong diagnosis made by doctors. Considering the technology is improving day by day, it is not easy to accept this issue since it can be solved with the power of technology. One medical area where people are more likely to face this problem is dentistry. Sometimes people lose their teeth because of the wrong diagnosis. However, by benefiting the technological advancement, such projects might be developed to make a diagnosis about the current situation of patient instead of doctor. In this case, this problem will mostly be solved.

The paper focuses on dental disease detection from X-Ray images of mouth using the advantages of Artificial Intelligence. Before diving into coding, let's dive into the previous research about object detection in X-ray images because so many similar projects have already been conducted.

**Related works**

As already underlined above, before starting any research projects, it is very crucial to check the related works to get some tips and tricks and decide about the path which should be followed during the project. Also, it is important to think about how to do something different and how to improve the existing work which has been done previously. Thus, several such works have been analyzed within the scope of internship and in the below part, each of them would be discussed.

D Suryani, M N Shoumi, and R Wakhidah write about dental disease detection in X-Ray image of mouth in their articles, “Object detection on dental x-ray images using deep learning method”. They emphasize the importance of dental disease detection in real life since the disease sometimes cannot be identified by doctors or they are misidentified. In their experiments, as a base deep learning model, R-CNN and its other versions such as Fast R-CNN, Faster R-CNN and Mask R-CNN have been used which we also discussed about those architectures in the following sections. They really achieved better scores as a result of their experiments which more than 90 %as an accuracy (Suryani et al., 2021). There is no need to dive more into this research and discuss the hyperparameters, dataset and so on they have used.

Another similar work done before has been discussed in the paper “Detection of Dental Diseases through X-Ray Images Using Neural Search Architecture Network” so that to identify tooth related problems, deep learning model has been built (AL-Ghamdi et al., 2022). The model which was built is NasNet convolution model and it was trained on 90% of the images. The remaining was kept for validation. It is also worth noting that the dental diseases that were being tried to detect were categorized based on three different classes, namely cavity, filling and implant. Besides the model itself, the authors have also emphasized in this paper the importance of data processing and augmentation. For this purpose, they have divided the project into two categories; one with data processing and augmentation, another without these steps. At the end of the paper, they have concluded that with data augmentation, more than 96% accuracy score was achieved whereas it was about 93% when the model was trained on pure dataset, without any pre-processing. The result of this research has been illustrated using a bar graph in Figure 1.



Figure 1. Comparison graph for the experiment with and without data augmentation

One more similar work has been published in International Journal of Environmental Research and Public Health journal this year (Tareq et al., 2023). In this research, the fifth version of YOLO algorithm family has been used. Thus, even one version of YOLO has several sub versions inside based on the model sizes. The articles compare them on the same use case which detects the cavity on the surface of the teeth as described in Figure 2. Besides the object detection algorithm itself, the article talks about the importance of data augmentation itself so that 13 different augmentation techniques have been used to enlarge the dataset which would be passed to YOLO model to be trained with.

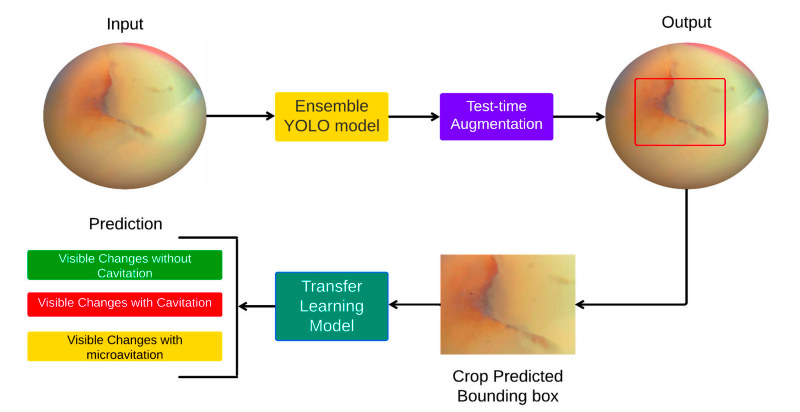


Figure 2. The flow of YOLO model application on teeth

Let's return to our project. The project consists of several sections which need to be explained separately in order to understand clearly what is going on. For instance, two different sequential CNN models have been applied in the project for two different purposes, one is for tooth segmentation out of the full X-ray image, the other is for detecting the teeth which have the specified disease. Thus, let's start with segmentation.

**Object Detection Algorithms**

Once the steps about segmentation are over, the next and one of the most important steps of the project is to apply object detection algorithm to classify the classes and draw bounding boxes around them. Certainly, in the world of Artificial Intelligence, so many object detection algorithms with different and complex architectures exist that should be analyzed carefully before applying them in real projects. Because their architectures were not defined and built just for fun, they were defined to solve a specific object detection issue; it might be to be able to detect very small objects or if the speed is more important, it might be designed to focus on the speed rather than the accuracy and so on. During the internship, several object detection algorithms have been analyzed in order to find the optimal one for our use case which needs to detect object in X-Ray images. Since the projects require to work with medical images, firstly R-CNN algorithm was analyzed, then Mask R-CNN, Detectron-2, respectively, and finally each version from first to eighth version of YOLO algorithm was analyzed. The information about each of them, especially their advantages and disadvantages would be clearly discussed in the upcoming sections, and then talked about the algorithm which we continue to work with as a result of this deep research. Considering the fact that each version of YOLO algorithm has a different backbone architecture, each of them should be accepted as a separate object detection algorithm. Thus, we may emphasize the fact during the selection process between different algorithms, more than ten algorithms have been analyzed by considering that some of them is new to the field of Artificial Intelligence which makes the research and decision process more difficult. The general information was given but to fully understand each of them, let's dive into the details about each algorithm.

**R-CNN (Region-Based Convolutional Neural Networks)**

In the search of the best object detection algorithm for our use case, the next algorithm that was analyzed is R-CNN which stands for Region-Based Convolutional Neural Networks. It is one the most popular algorithms in the field of computer vision in terms of its better performance for detecting objects. While detecting the object in the image, the R-CNN algorithm applies a two-stage approach. In the first stage, it uses selective search algorithm using set of ROIs (Region of Interest) so that each ROI is trying to find whether the object is present in the image or not. Thus, ROIs are responsible to extract and form the feature map from the input image which would be passed to Convolutional Neural Network in the next stage. Once CNN is fed with the feature vectors, it starts to classify the object using the information presented.

While discussing each algorithm, it is important to emphasize their pros and cons in order to show why this algorithm was selected or eliminated to use in the project. In terms of the advantages of the R-CNN algorithm, its remarkable accuracy and object localization power should be underlined. Under several observations in object detection tasks, it might be concluded that R-CNN achieves a high accuracy score and draws precise bounding box around the object which is the one that needs to be detected. Certainly, all these performances require a better prepared dataset to feed the model. On the other hand, the model requires too much time to train and has low speed to detect the object because of the complex architecture that the model has. As the complex architecture, region proposal network can be underlined so that processing each region in the image separately and forming the separate feature maps which will be passed to CNN makes the model to work very slowly and requires high computational resources to perform and calculate all these operations (*Proposing a CNN Method for Primary and Permanent Tooth Detection and Enumeration on Pediatric Dental Radiographs - CORE Reader*, n.d.).

**Mask R-CNN**

Considering the limitations of R-CNN, Mask R-CNN was developed to overcome them. Besides detecting the object, Mask R-CNN also applies pixel-level mask for each instance. As a result of it, the algorithm achieves state-of-art performance in the field of object detection and computer vision. As it is noted that this algorithm was designed to improve the performance of R-CNN, the model architecture was also simplified in order to gain additional speed during the training. Thus, the model can perform end-to-end training in terms of simplified architecture between mask stage and CNN layers. However, all the improvements that have been made so far affect negatively to the model on the other hand. Obviously, additional pixel-level masking will require more resources to be performed as well as the training would be slow even though simplified architecture (Aparna et al., 2021). As a result, these two models, namely R-CNN and Mask R-CNN will not be able to work in real time because of their low speed.

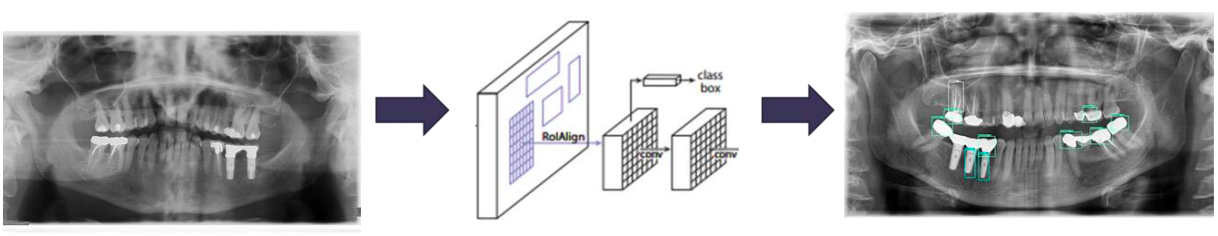


Figure 6. The general flow of object detection using CNN model

**YOLO**

As a result of the discussion above, we have already got familiar with two object detection algorithms and understand their strong and weak side. Let's have a look at another powerful abject detection algorithm which grows so rapidly that already has eight different versions. This algorithm is called YOLO which stands for “You Only Look Once” and as can be understood from the name, the algorithm is very powerful in real-time object detection use cases since it is the type of one stage object detection algorithm whereas R-CNN is two stage which make it slower compared with YOLO (Diwan et al., 2022).

In general, the architecture of the versions of YOLO algorithm contains three different parts, namely backbone, head and neck where each of them has special task to perform (Zhao et al., 2022). These components have been visualized in Figure 7.

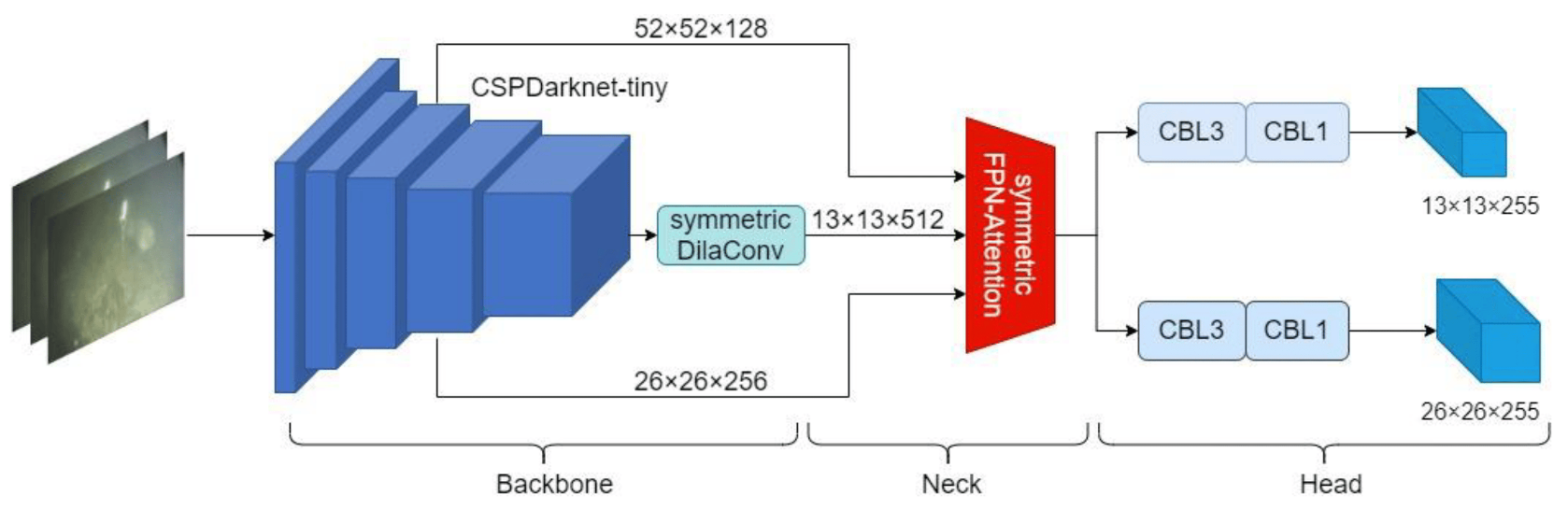


Figure 7. The general architecture of initial YOLO model

Thus, the backbone is responsible for extracting the important features which later passed to the neck. Neck is the second component which creates feature pyramids over the crucial features which are collected by the backbone. Once the feature pyramids are ready, they are going to the head for final detection (*YOLO Algorithm for Object Detection Explained [+Examples]*, n.d.).

As noted, YOLO has eight versions where in each new version, the previous version has been improved. To have a better understanding about each version and get more information about those enhancements and improvements, let's look at each version separately.

**YOLOv1**

YOLOv1 is the first version of the YOLO object detection algorithm. It introduced a groundbreaking approach to object detection by dividing the input image into a grid and associating each grid cell with predicted bounding boxes and class probabilities. YOLOv1 employed a single neural network to simultaneously predict object classes and bounding box coordinates in a single pass.

YOLOv1 contains two crucial components: the feature extraction network and the detection network. The feature extraction network, called Darknet, is a deep convolutional neural network that takes the image as input, process it and takes high-level features out of the images processed. The detection network accepts these features and makes predictions for bounding boxes and class probabilities.

To draw bounding boxes, YOLOv1 divides the image into an S x S grid. Every grid cell is responsible for identifying bounding boxes along with their corresponding accuracy scores. Each box is characterized by 5 parameters: the x and y coordinates, width and height, and the confidence score demonstrating the probability that the bounding box stores an object (Figure 8). Class probabilities are also predicted for each grid cell, indicating the probability of each class being present within that cell (*YOLO Algorithm: Real-Time Object Detection From a to Z*, n.d.).

During training, YOLOv1 uses labeled bounding box annotations to calculate the loss function, which includes localization loss (measuring the accuracy of predicted bounding boxes) and classification loss (measuring the accuracy of predicted class probabilities). The loss is backpropagated through the network to update the weights and improve the model's performance.

YOLOv1 achieved impressive results by predicting object classes and bounding box coordinates in a single pass, making it highly efficient. It demonstrated real-time performance and competitive accuracy on various datasets. However, YOLOv1 had limitations in accurately localizing small objects and handling overlapping objects due to its grid-based approach. Despite these limitations, YOLOv1 was a pioneering algorithm that laid the foundation for subsequent versions of the YOLO series.

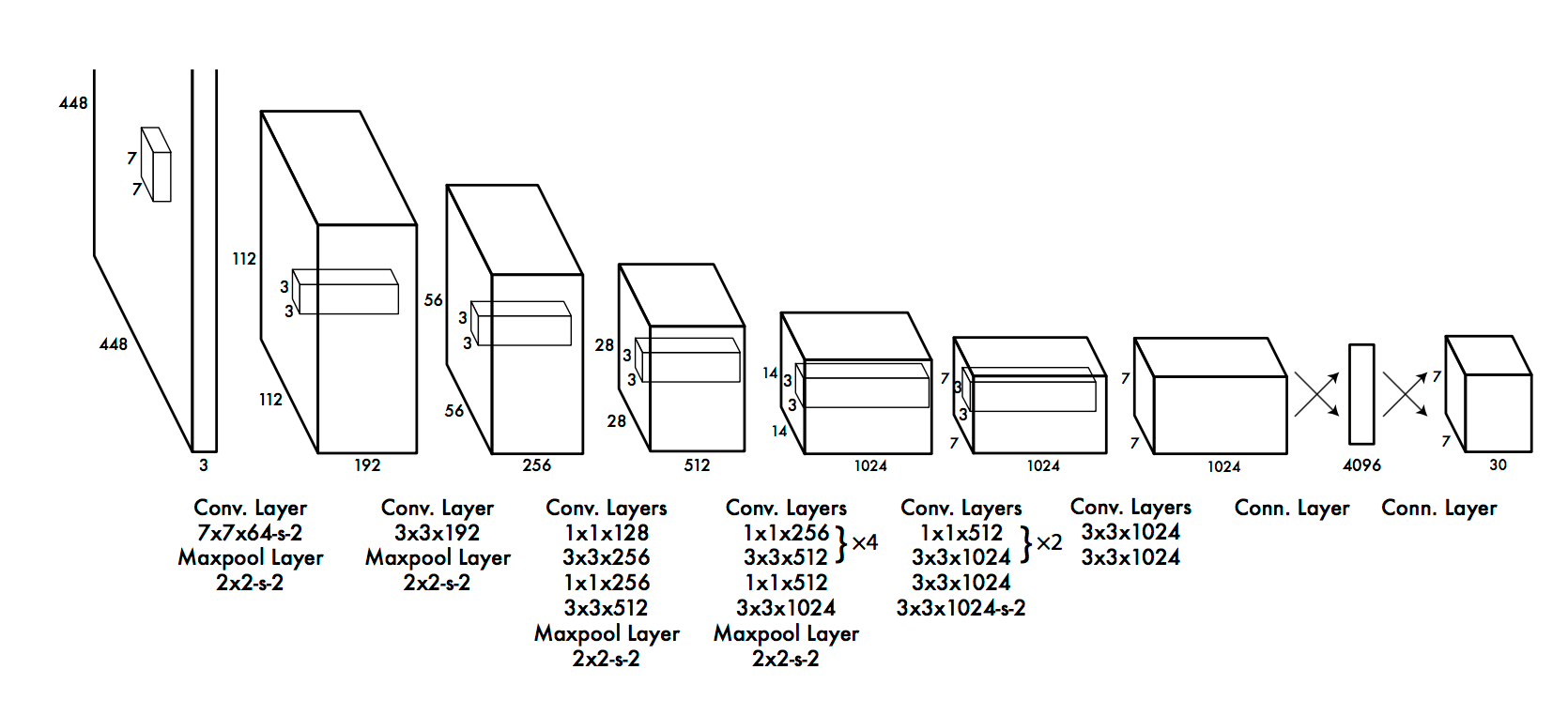


Figure 8. Backbone architecture of YOLOv1

**YOLOv2**

YOLOv2, also known as YOLO9000, is the second version of the YOLO object detection algorithm. It introduced significant improvements over YOLOv1 to enhance accuracy and address its limitations. Here's an overview of YOLOv2:

YOLOv2 introduced several key enhancements to the original YOLO framework. One major improvement is the incorporation of anchor boxes. Instead of predicting bounding box coordinates directly, YOLOv2 predicts offsets to anchor boxes of different shapes and sizes. These anchor boxes serve as reference templates for object detection and help improve localization accuracy.

YOLOv2 also implemented a concept called multi-scale training and testing. The network is trained on images of different sizes, enabling it to detect objects at various scales effectively. During testing, YOLOv2 performs detection on multiple scales, ensuring the detection of objects of different sizes in the input image. To improve the specified network architecture, YOLOv2 introduced a custom deep neural network called Darknet-19. This network, with 19 convolutional layers, extracts feature representations from the input image, providing a complex context for object detection (Figure 9). Darknet-19 is designed specifically for object detection tasks and contributes to improved performance.

YOLOv2 extended the object detection abilities by incorporating a large-scale object detection and classification dataset called COCO (Common Objects in Context). By training on COCO, YOLOv2 can detect and classify a broader range of objects beyond the classes in the original YOLO dataset. Another core side of YOLOv2 is its ability to perform joint training on many datasets, allowing the model to capture patterns from different sources of data. This approach leads to better generalization and robustness of the model.

Overall, YOLOv2 showed significant developments in accuracy and addressed some of the drawbacks of YOLOv1. It got competitive performance while performing real-time processing speed. YOLOv2 laid the foundation for subsequent versions, paving the way for further developments in the YOLO series (Hui, 2022).

Diagram

Description automatically generated

Figure 9. YOLOv2 Network Architecture

**YOLOv3**

YOLOv3 is the third version of the YOLO object detection algorithm. It builds by relying on the strengths of its predecessors while introducing several key developments to improve object detection performance. Here's an overview of YOLOv3:

YOLOv3 introduced a larger and deeper network architecture called Darknet-53. Darknet-53 consists of 53 convolutional layers (Figure 10), making it more powerful and capable of capturing complex features. The complicated model architecture allows for better training and feature extraction, resulting in improved detection accuracy. One of the significant advancements in YOLOv3 is the introduction of feature pyramid scales. YOLOv3 utilizes feature maps at three different scales (referred to as "YOLOv3-tiny," "YOLOv3-small," and "YOLOv3-large") to detect objects of varying sizes. By identifying objects at multiple shapes, YOLOv3 can handle objects of different sizes more effectively, improving detection performance across the entire image.

YOLOv3 also introduced the ability to predict at multiple resolutions so that YOLOv3 predicts objects at three different resolutions within each scale. This multi-resolution prediction ability further improves the detection capabilities and makes the model to get objects with different levels of granularity. To tackle the challenge of accurately localizing small objects, YOLOv3 utilizes a technique called "skip connections." These connections allow information from earlier layers to be directly propagated to later layers, enabling the model to capture both low-level and high-level features. This helps improve the localization accuracy of small objects. Besides all these, YOLOv3 incorporates several other optimizations to enhance performance, including the use of anchor boxes for bounding box predictions, the adoption of the "route" layer to concatenate feature maps from different layers, and the implementation of the "upsample" layer for upsampling feature maps.

By combining these improvements, YOLOv3 achieved state-of-the-art performance in terms of accuracy while maintaining real-time processing speed. It demonstrated remarkable object detection capabilities, especially for detecting objects of different sizes and dealing with complex scenes.

YOLOv3 has found applications in various domains, including autonomous driving, surveillance systems, and robotics, where real-time object detection is crucial. Its advancements have paved the way for further improvements in subsequent versions of the YOLO series (Kathuria, 2018).

Diagram

Description automatically generated

Figure 10. The architecture of YOLOv3 model

**YOLOv4**

YOLOv4 is the fourth version of the YOLO object detection algorithm. It represents a significant advancement over its predecessors, incorporating several innovative techniques to further improve accuracy and performance. YOLOv4 introduced a new network architecture called "CSPDarknet53" (Cross-Stage Partial Darknet53). CSPDarknet53 is designed to enhance feature representation by incorporating Cross Stage Partial (CSP) connections (Figure 11). These connections allow for improved information flow between different layers, enabling more effective feature extraction and enhancing object detection performance.

One of the key enhancements in YOLOv4 is the adoption of the Mish activation function. Mish is an alternative activation function that has been shown to provide better gradient flow, resulting in improved training stability and performance. Mish helps YOLOv4 capture more nuanced features and improves the model's ability to detect objects accurately.

YOLOv4 introduced various techniques to address the challenge of handling imbalanced classes during training. It utilizes a modified version of the focal loss function, called the "YOLOv4 Loss," which helps to mitigate the impact of class imbalance and focuses on hard-to-detect objects. This modification improves the model's ability to handle object detection tasks with imbalanced datasets.

To further improve the detection performance, YOLOv4 applied advanced data augmentation techniques. One important technique is mosaic data augmentation, where several images are gathered into a mosaic during training. This augmentation strategy increases the variety of training samples, leading to good generalization and enhanced detection results. YOLOv4 incorporates architectural enhancements such as PANet (Path Aggregation Network) and SAM (Spatial Attention Module). PANet helps to aggregate features at different dimensions, allowing for improved representation of feature and object detection through many resolutions. SAM focuses on getting spatial dependencies within feature maps, enhancing the model's ability to identify objects accurately.

Additionally, YOLOv4 came with several optimization techniques to increase training and testing speed. It utilizes the "Darknet" framework, which is well-optimized for GPU acceleration. YOLOv4 also benefits from advancements in hardware like Tensor Cores and mixed-precision training, enabling faster and more efficient computations (Ji et al., 2023).

Overall, YOLOv4 represents a significant leap in object detection performance. It achieves state-of-the-art accuracy while maintaining real-time or near-real-time processing speed. YOLOv4 has been widely adopted in applications such as autonomous driving, video surveillance, and object recognition, where accuracy and speed are critical. Its advancements have contributed to pushing the boundaries of real-time object detection capabilities.

Chart, waterfall chart

Description automatically generated

Figure 11. YOLOv4 model architecture

**YOLOv5**

YOLOv5 (You Only Look Once) is a popular and recent version of the YOLO object detection algorithm. Developed as an independent project, YOLOv5 introduces several improvements and optimizations over previous versions. YOLOv5 aims to strike a balance between accuracy and inference speed by focusing on lightweight architecture. The model architecture of YOLOv5 consists of a convolutional backbone network, a feature pyramid network (FPN), and detection heads. The backbone network extracts high-level features, while the FPN enhances feature representation at different scales for improved object detection.

One of the key improvements in YOLOv5 is the implementation of a more efficient and streamlined architecture compared to its predecessors. YOLOv5 achieves this by employing a "CSPDarknet" backbone, which is a variant of the Darknet architecture used in previous versions (Figure 12). The CSPDarknet architecture improves training and inference efficiency, enabling real-time performance even on resource-constrained devices.

Additionally, YOLOv5 utilizes a wide range of data augmentation techniques, including mosaic augmentation, random scaling, and rotation, to diversify the training data and improve the model's ability to generalize to different scenarios.

One notable feature of YOLOv5 is its user-friendly and flexible design. The YOLOv5 project provides a user-friendly command-line interface (CLI) and a comprehensive set of configuration options. This allows users to easily customize and fine-tune the model for their specific application requirements. The YOLOv5 ecosystem also includes pre-trained models and tools for data preparation, training, and evaluation. YOLOv5 offers different model sizes (Yolov5s, Yolov5m, Yolov5l, Yolov5x), allowing users to choose the appropriate trade-off between speed and accuracy based on their specific use case. Smaller models like Yolov5s prioritize faster inference, while larger models like Yolov5x provide higher accuracy at the cost of increased computational requirements. While YOLOv5 has gained popularity and demonstrated impressive results on various object detection tasks, it is important to note that it is not an official release from the original YOLO creators. YOLOv5 has been developed as an independent project by a different team, leveraging ideas and techniques from the YOLO series.

In summary, YOLOv5 is a lightweight and efficient version of the YOLO object detection algorithm. It combines a streamlined architecture, transfer learning, and advanced data augmentation to achieve a good balance between accuracy and inference speed. YOLOv5 offers flexibility, ease of use, and a range of model sizes to suit different application requirements.

Diagram

Description automatically generated

Figure 12. YOLOv5 model architecture

**YOLOv6**

The next release of YOLO family is YOLOv6 which is built with different CNN architecture and has different version as well with different model size including YOLOv6-n, YOLOv6-tiny and YOLOv6-s. The architecture that has been used in YOLOv6 is EfficientNet-L2 which is the type of EfficientNet architectures (Figure 13). The experiments also show that the speed has also been improved with this new version of YOLO algorithms. Thus, compared with YOLOv5, the sixth version has 3.59 ms latency while it was 5.38 ms in fifth version (2022). The reduction in latency is also related to the fact that YOLOv6 architecture has been designed to be able to work well with GPU. The picture below illustrates the neck and backbone architecture of YOLOv6 algorithm.

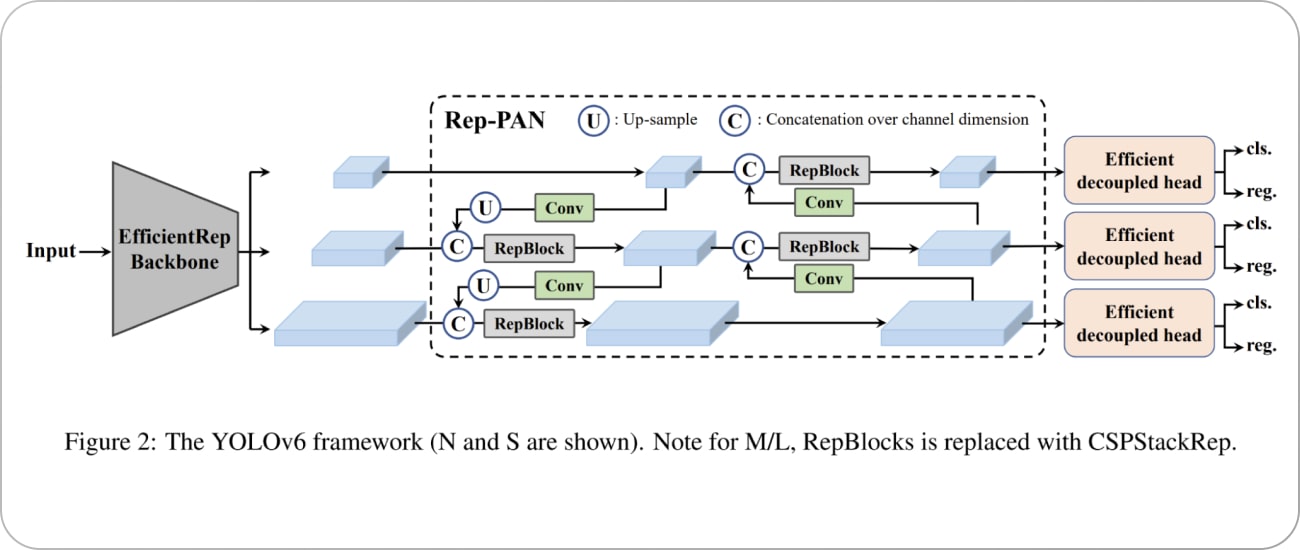


Figure 13. YOLOv6 model architecture

**YOLOv7**

In the search for the fastest and more accurate object detection algorithm, YOLOv6 is optimized more and as a result, YOLOv7 was developed. Based on the model size, several YOLOv7 models exist such as YOLOv7-Tiny, YOLOv7-W6, YOLOv7-X and so on. In this version, E-ELAN (Extended Efficient Layer Aggregation Network) has been used as a backbone network which is concatenation-based network. Moreover, compared with the previous version, YOLOv7 requires 40% less parameters which results in higher computational efficiency. As a result, the model is able to be trained in very less time compared with other deep learning models. Also, experiments show that in this version, high accuracy has been obtained (*YOLOv7 Paper Explanation: Object Detection and YOLOv7 Pose*, 2022).

**Detectron2**

As it was underlined at the beginning, as a last object detection algorithm, Detectron2 was analyzed to decide whether it would better suit our use case or not. For that reason, let's have a look at the architecture of this algorithm and be aware of how it works. Then, we may check an experiment where Detectron2 was compared with YOLO in order to compare their performances.

Detectron2 is another deep learning model for object detection that was developed on PyTorch library by Facebook AI Research. It has a bit more complex architecture which will be discussed now. The architecture contains several reusable components like backbone network, region proposal network and feature extractors that work together to perform object detection. These components can work as a whole, or one can be replaced with another. Thus, Detectron2 supports different backbone networks such as ResNet, MobileNet and so on which works as feature extractors (Shaikh, 2021). Once the feature maps are extracted, Detectron2 has RPNs (Region Proposal Network) to draw bounding boxes around the object. The algorithm architecture has also ROI Pooling or ROIAlign which is a technique that extracts fixed-size feature maps from the regions proposed by the RPN. The flow of algorithms has been visualized in Figure 14. In general, Detectron2 has gained popularity due to its flexibility, modular design, and impressive performance on various benchmark datasets. Its rich set of features and easy-to-use APIs make it a preferred choice for researchers and practitioners working on object detection tasks (Honda, 2022).

**DEVELOPMENT**

Now let's have a look at the experiment done during the internship about dental disease detection in X-Ray images. The AI project actually contains several sections so that we need to explain them separately to understand what is going on. The first and one the most important part of the project is about dataset. It was already discussed about the dataset for image segmentation part at the beginning. For the disease detection part, we will use those segmented images as an input for the deep learning model which YOLOv5. For this purpose, we have 116 images in the dataset which have already been segmented and they have corresponding labels stored in XML files in another directory. However, the XML files have special content and store too much information which we need to extract the data we need and parse to TXT file, as a first step. But in general, the XML files have the information about the position of the disease which will be used to draw rectangular shape around the teeth, and it has information about the disease itself. Initially, there are eight different categories of illness but in this project, three of them will be used which are Cavity, Radiolucent lesion and Radiopaque lesion (Figure 17).

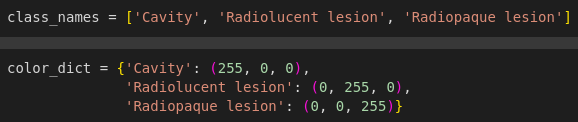


Figure 17. Class names

Thus, while parsing the information, there should be a filter based on these three categories. Another important point is that not all images have the corresponding XML files so that we need to filter the images as well and keep only those which have the labels because the model training process requires the labels to be present to pass to the model together with the images. Thus, we have already got familiar with the dataset available. Now, let's move to the next part which is about turning this dataset into more useable form so that it can be used to feed the model.

Once we have filtered all the images which have the corresponding labels in the xml files, we will move to work with the labels. As noted above, xml files store too much information which should be parsed to txt files. On the other hand, the model requires txt files which have pre-defined structure so that the content of txt file should follow this structure: class label (converted to an integer) and the position of rectangular shape (minimum and maximum values for each x and y dimension). Thus, while parsing xml files, this structure should be taken into consideration, otherwise, it will make additional work later. Let's show the steps in the code and its output. Initially, the information from xml files was taken as in Figure 18.



Figure 18. Class labels

Then, these data were written to txt file in such a structure that already discussed above. It should also be considered that the name of text file should be the same as the corresponding image. Figure 19 illustrates the process to save data in the text file.

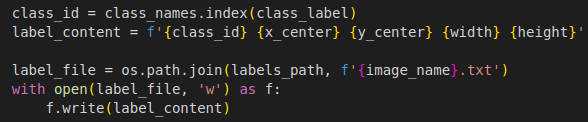


Figure 19. Parsing data from xml format to txt format

Let`s have a look at one example label in Figure 20.

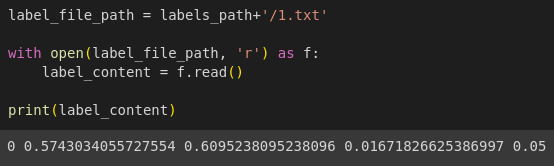


Figure 20. Example txt file

As can be clearly seen from the example text file, the file contains the label and its positions.

After all txt files were created, the next step is to create training and validation dataset to feed the model. YOLO accepts the dataset as the path which contains the images and corresponding text files which have the same name as the image itself. The whole dataset was taken and divided into 80% as training and the remaining 20% as validation dataset, and they have been stored in two different directories. And all this information will be stored in the configuration file which is data.yaml. Thus, the yaml file will store the path to the training and validation dataset, number of classes that we are trying to identify and the names of those classes. Figure 21 describes the content of yaml file:

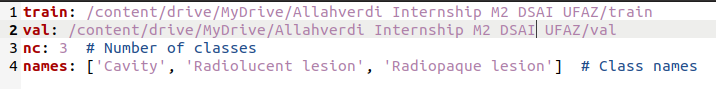


Figure 21. The content of Yaml file

Once the dataset was ready, we needed to move to work with the model itself which is the fifth version of YOLO family so that the model should be installed. With the command below, the model will use the configuration information in the yaml file, define the epoch and batch size. (Figure 22)



Figure 22. Train the YOLOv5 model

After the command is executed, based on the information provided such as yaml content, image size, batch size, epoch and so on the model will be trained and the weights will be stored in the directory **yolov5/runs/train/exp6.** The model summary is described in Figure 23 which helps to get more information about its architecture (the number of layers used), optimizer (such as learning rate) and so on.

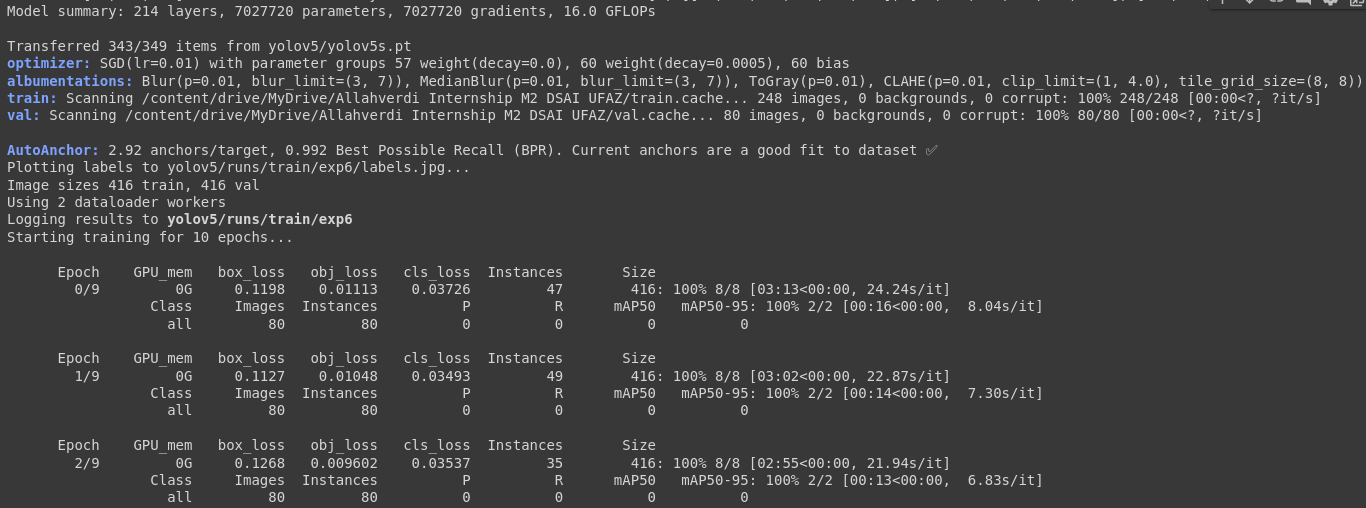


Figure 23. The information about model architecture and

**RESULTS & DISCUSSION**

Thus, the model is trained and ready to be used. It was saved in the specified directory so that it can be loaded into jupyter notebook and used for professional purposes. The below image describes how the scores vary with the epoch and the final scores of the model. Finally, I will attach those informational images below so that everything will be clearer about the labels, training steps, scores and so on.

In the figure 24, the trained classes distribution has been illustrated. Thus, it is more clear from the figure that how many samples from each classes have been passed to YOLO model for disease detection. Moreover, there are two scatter plot on the same figure where there is information about x and y relationship as well as the relation between width and height. As already discussed before, these values which are x, y, width and height were passed to the model within txt file which store the annotations for each classes and their positions.

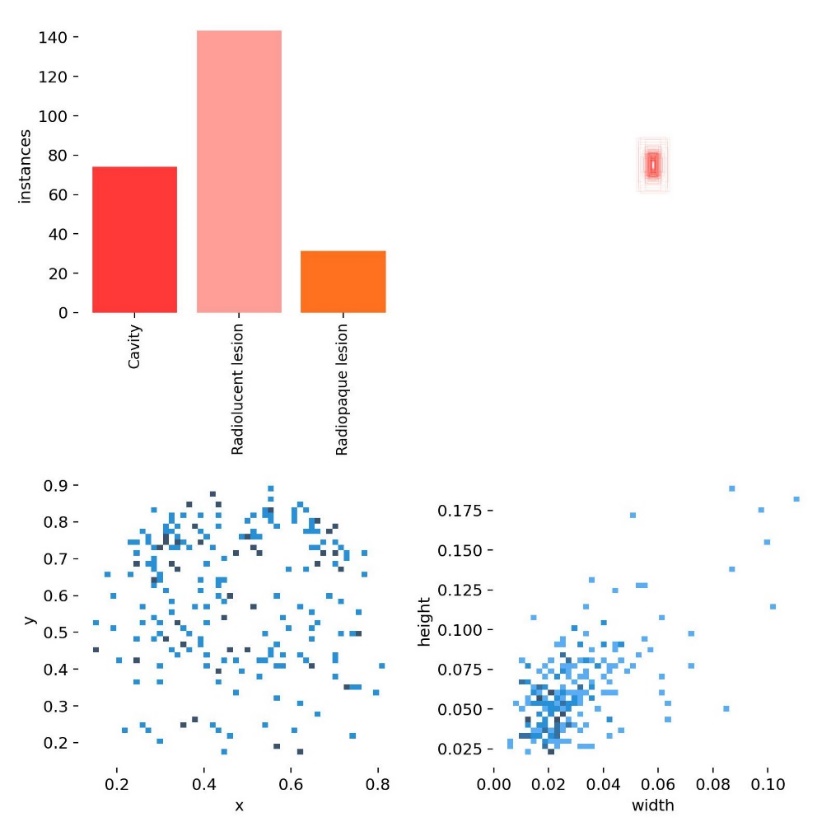


Figure 24. Classes and their distribution

Figure 25 gives information about the correlation between all 4 numeric parameters which are the positions of bounding boxes. From the correlation matrix, it is easier to see the distribution and the relation of different parameters with each-others.

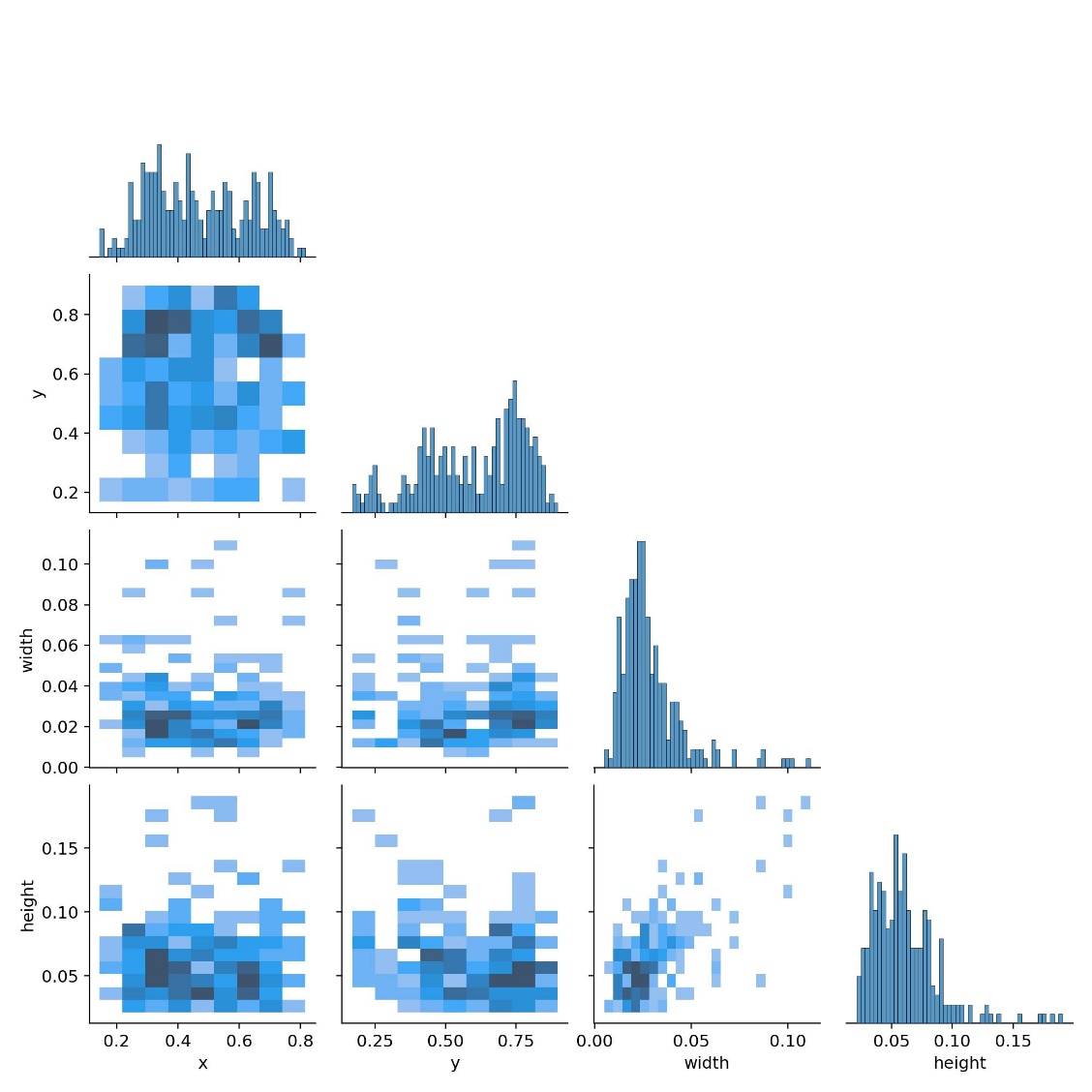


Figure 25. Labels correlations

The YOLOv5 model accepts several hyper parameters which should be tested with different values in order to get the optimal ones for the specific use case. That`s why during the experiment, I have too many cases which were failed or did not end as expected as a result of those parameters. Taking the resources of computer in which model will be trained into account, I have carefully set the parameters so that I started from small numbers then increased them. The final Yolo model trained for 200 epochs, and each epoch took 48 sample image. Each image size was 640, and learning rate parameter was set to 0.01. Besides those parameters, different augmentation techniques such as flipping, cropping and so on were used, and their values were tested many times. Figure 26 illustrates the line graphs where we can get information about how the box loss, precision, recall and mean average precision (mAP) score varies within each epoch. I got 80% mAP score which was the best score among all test cases. Precision score was above 90% and recall score was very close to 90%. As it should be, the box loses in both training and validation dataset is going down in the next epoch.

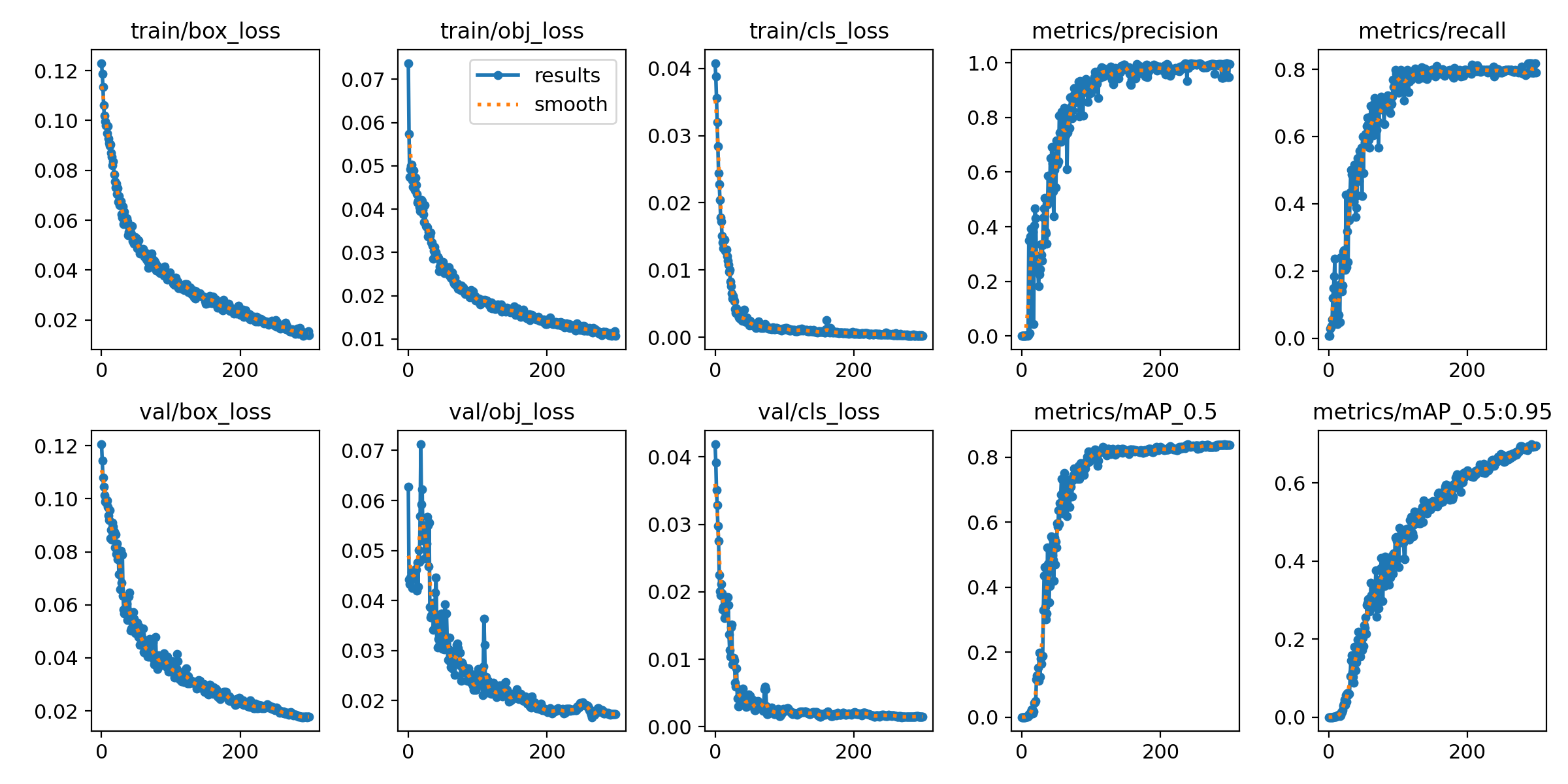


Figure 26. Evaluation metrics

Once the model was trained and ready to use, I tested it on a single image which has several diseases including the ones that we trained the model with. The output was very close to the real so that the model was able to identify the disease with probability more than 61%. Figure 27 is the output of this prediction.

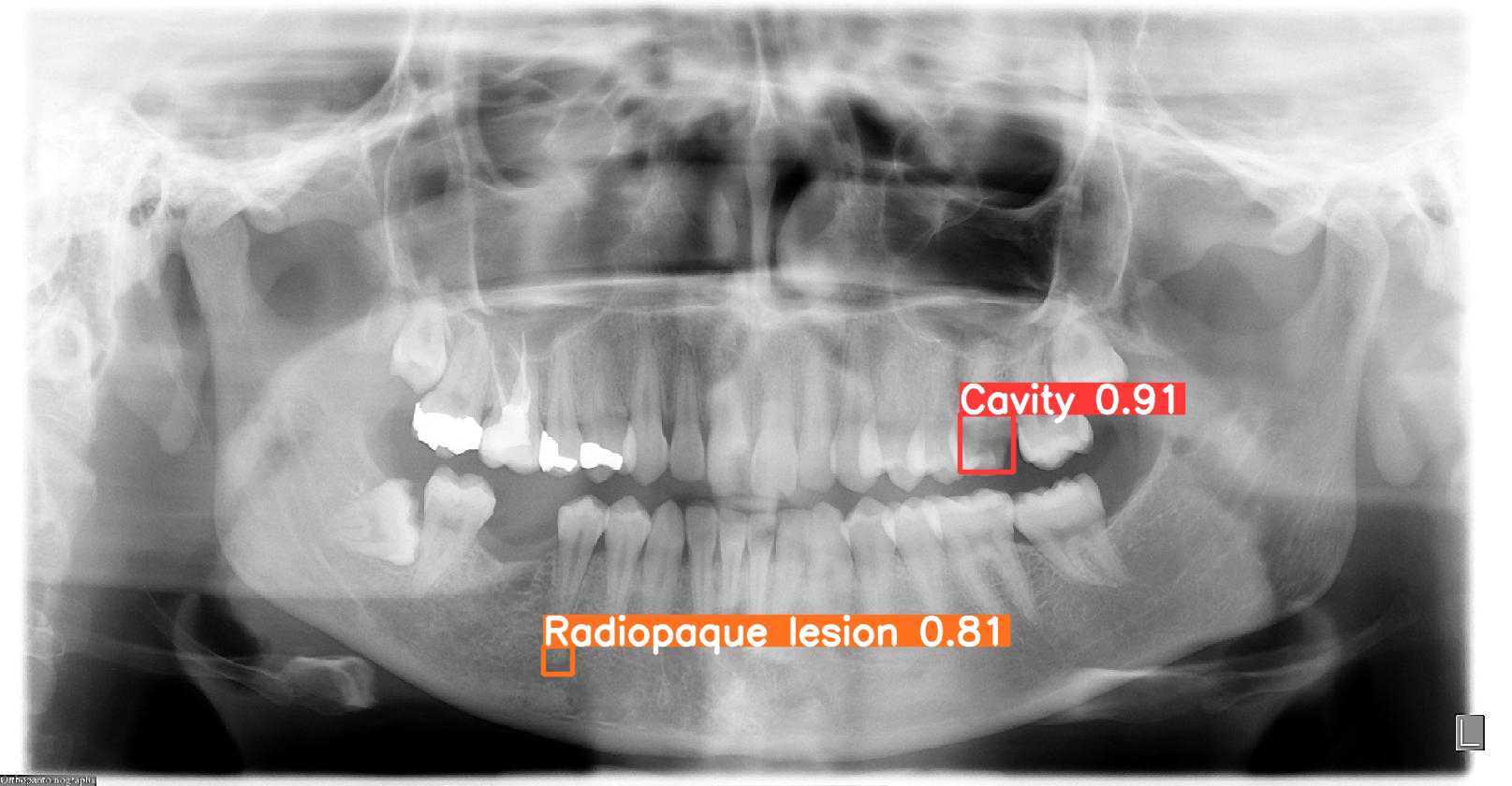
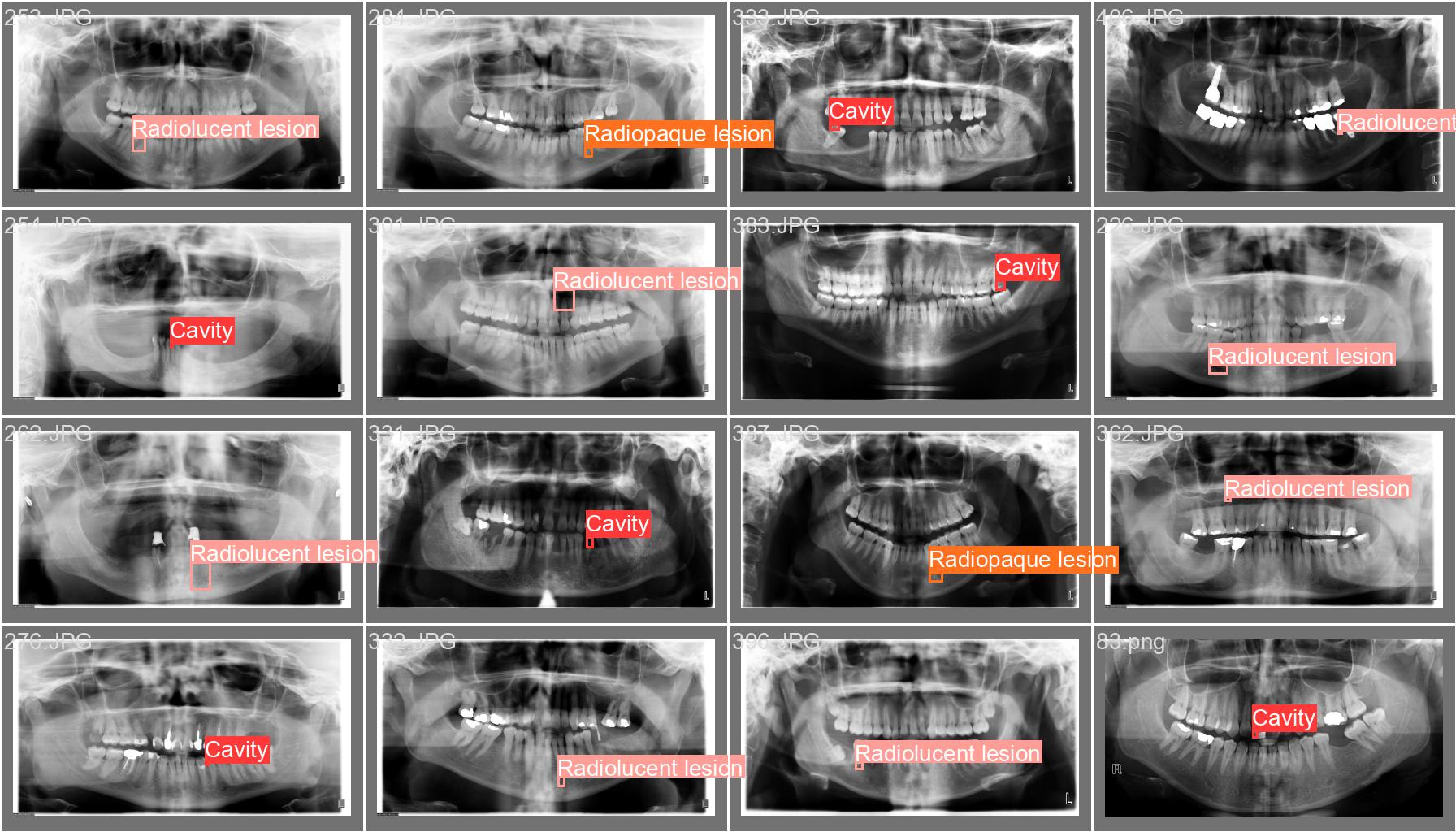


Figure 27. Model testing



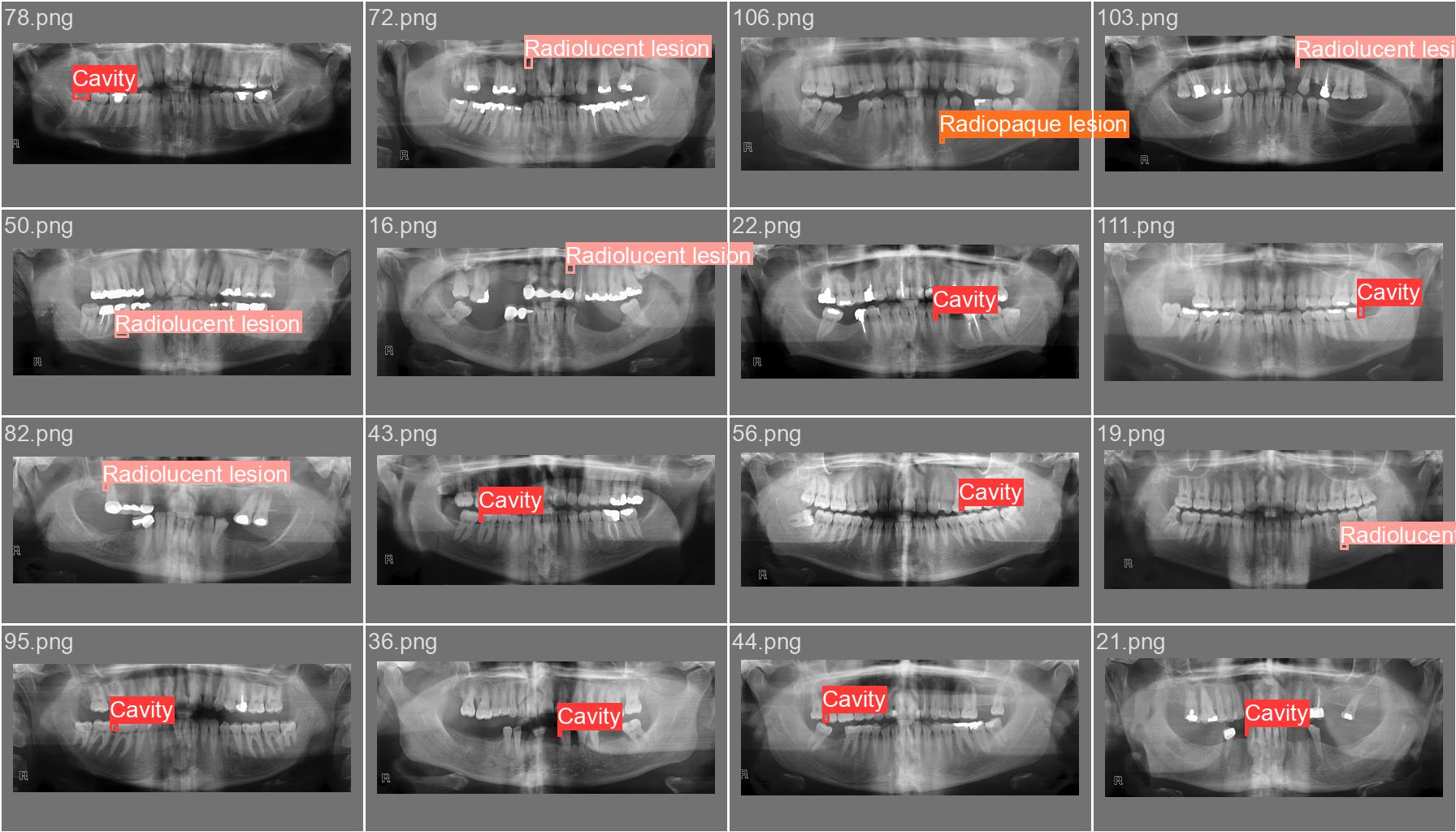


Figure 28. Training batch

**CONCLUSION**

To sum up all the work done until now, we were trying to build a model for dental disease detection in X-Ray images. To get the better model, different techniques and research were applied, and all were clearly discussed in the paper. We conducted a research to find the best object detection algorithm for our use case since many of them exist in the field of Artificial Intelligence. As already discussed above, several algorithms were analyzed and checked whether it suits our use case or not. Finally, the fifth version of YOLO algorithms was selected based on its performance and other advantages which were underlined. Finally, all these techniques and research gave us direction and helped to achieve really good results in the end.

**REFERENCES**

1. AL-Ghamdi, A. S. A. M., Ragab, M., AlGhamdi, S. A., Asseri, A. H., Mansour, R. F., & Koundal, D. (2022, April 30). Detection of Dental Diseases through X-Ray Images Using Neural Search Architecture Network. *Computational Intelligence and Neuroscience*, *2022*, 1–7. <https://doi.org/10.1155/2022/3500552>
2. Suryani, D., Shoumi, M. N., & Wakhidah, R. (2021, February 1). Object detection on dental x-ray images using deep learning method. *IOP Conference Series: Materials Science and Engineering*, *1073*(1), 012058. <https://doi.org/10.1088/1757-899x/1073/1/012058>
3. Tareq, A., Faisal, M. I., Islam, M. S., Rafa, N. S., Chowdhury, T., Ahmed, S., Farook, T. H., Mohammed, N., & Dudley, J. (2023, March 31). Visual Diagnostics of Dental Caries through Deep Learning of Non-Standardised Photographs Using a Hybrid YOLO Ensemble and Transfer Learning Model. *International Journal of Environmental Research and Public Health*, *20*(7), 5351. <https://doi.org/10.3390/ijerph20075351>
4. Almalki, Y. E., Din, A. I., Ramzan, M., Irfan, M., Aamir, K. M., Almalki, A., Alotaibi, S., Alaglan, G., Alshamrani, H. A., & Rahman, S. (2022, September 28). Deep Learning Models for Classification of Dental Diseases Using Orthopantomography X-ray OPG Images. *Sensors*, *22*(19), 7370. <https://doi.org/10.3390/s22197370>
5. Saxena, S., Thorat, S., Jain, P., Mohanty, R., & Baraskar, T. (2022, August 1). Analysis of object detection techniques for bird species identification. *Journal of Physics: Conference Series*, *2325*(1), 012054. <https://doi.org/10.1088/1742-6596/2325/1/012054>
6. Wang, M., Yang, B., Wang, X., Yang, C., Xu, J., Mu, B., Xiong, K., & Li, Y. (2022, November 21). YOLO-T: Multitarget Intelligent Recognition Method for X-ray Images Based on the YOLO and Transformer Models. *Applied Sciences*, *12*(22), 11848. <https://doi.org/10.3390/app122211848>
7. Dumitrescu, F., Ceachi, B., Truică, C. O., Trăscău, M., & Florea, A. M. (2022, September 17). A Novel Deep Learning-Based Relabeling Architecture for Space Objects Detection from Partially Annotated Astronomical Images. *Aerospace*, *9*(9), 520. <https://doi.org/10.3390/aerospace9090520>
8. Li, Z., Tian, X., Liu, X., Liu, Y., & Shi, X. (2022, January 14). A Two-Stage Industrial Defect Detection Framework Based on Improved-YOLOv5 and Optimized-Inception-ResnetV2 Models. *Applied Sciences*, *12*(2), 834. <https://doi.org/10.3390/app12020834>
9. Saxena, S., Thorat, S., Jain, P., Mohanty, R., & Baraskar, T. (2022, August 1). Analysis of object detection techniques for bird species identification. *Journal of Physics: Conference Series*, *2325*(1), 012054. <https://doi.org/10.1088/1742-6596/2325/1/012054>
10. *Introduction to YOLO Algorithm for Object Detection*. (n.d.). Engineering Education (EngEd) Program | Section. <https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/>
11. *YOLOv7 Paper Explanation: Object Detection and YOLOv7 Pose*. (2022, August 2). LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow with Examples and Tutorials. <https://learnopencv.com/yolov7-object-detection-paper-explanation-and-inference/>
12. D. (2022, November 2). *How Does YOLOv6 Differ From YOLOv5 or YOLOX? - Deci*. Deci. <https://deci.ai/blog/yolov6-vs-yolov5-vs-yolox/>
13. Mukherjee, A. (2022, December 8). *Multi-class wildlife classification using YOLOv5, YOLO v7 and Detectron2- Faster RCNN*. Medium. [https://medium.com/@amritangshu.mukherjee/multi-class-wildlife-classification-using-yolov5-yolo-v7-and-detectron2-faster-rcnn-b8423ffdbf6b](mailto:https://medium.com/@amritangshu.mukherjee/multi-class-wildlife-classification-using-yolov5-yolo-v7-and-detectron2-faster-rcnn-b8423ffdbf6b)
14. *YOLO Algorithm for Object Detection Explained [+Examples]*. (n.d.). YOLO Algorithm for Object Detection Explained [+Examples]. <https://www.v7labs.com/blog/yolo-object-detection>
15. Shaikh, T. (2021, October 12). *Object detection in six steps using Detectron2*. Data, AI & Product Engineering | Algoscale. <https://algoscale.com/blog/object-detection-in-six-steps-using-detectron2/>
16. Honda, H. (2022, January 18). *Digging into Detectron 2*. Medium. <https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd>
17. *YOLO Algorithm: Real-Time Object Detection from A to Z*. (n.d.). Kili-website. <https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z>
18. Diwan, T., Anirudh, G., & Tembhurne, J. V. (2022, August 8). Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, *82*(6), 9243–9275. <https://doi.org/10.1007/s11042-022-13644-y>
19. Ji, S. J., Ling, Q. H., & Han, F. (2023, January). An improved algorithm for small object detection based on YOLO v4 and multi-scale contextual information. *Computers and Electrical Engineering*, *105*, 108490. <https://doi.org/10.1016/j.compeleceng.2022.108490>
20. Zhao, S., Zheng, J., Sun, S., & Zhang, L. (2022, August 11). An Improved YOLO Algorithm for Fast and Accurate Underwater Object Detection. *Symmetry*, *14*(8), 1669. <https://doi.org/10.3390/sym14081669>
21. Hui, J. (2022, September 6). *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3*. Medium. <https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088>
22. Kathuria, A. (2018, April 29). *What’s new in YOLO v3?* Medium. <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>
23. *Proposing a CNN method for primary and permanent tooth detection and enumeration on pediatric dental radiographs - CORE Reader*. (n.d.). Proposing a CNN Method for Primary and Permanent Tooth Detection and Enumeration on Pediatric Dental Radiographs - CORE Reader. <https://core.ac.uk/reader/539616490>
24. 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>
25. Aparna, S., Muppavaram, K., Ramayanam, C. C. V., & Ramani, K. S. S. (2021). Mask RCNN with RESNET50 for Dental Filling Detection. *International Journal of Advanced Computer Science and Applications*, *12*(10). <https://doi.org/10.14569/ijacsa.2021.0121079>
26. Luo, D., Zeng, W., Chen, J., & Tang, W. (2021, December 13). Deep Learning for Automatic Image Segmentation in Stomatology and Its Clinical Application. *Frontiers in Medical Technology*, *3*. <https://doi.org/10.3389/fmedt.2021.767836>
27. O’Sullivan, C. (2023, March 8). *U-Net Explained: Understanding its Image Segmentation Architecture*. Medium. <https://towardsdatascience.com/u-net-explained-understanding-its-image-segmentation-architecture-56e4842e313a>
28. Du, G., Cao, X., Liang, J., Chen, X., & Zhan, Y. (2020, March 1). Medical Image Segmentation based on U-Net: A Review. *Journal of Imaging Science and Technology*, *64*(2), 20508–1. <https://doi.org/10.2352/j.imagingsci.technol.2020.64.2.020508>

**LIST OF ABBREVIATIONS AND CONVENTIONAL SIGNS**

|  |  |
| --- | --- |
| **Abbreviation** | **Definition** |
| YOLO | You Only Look Once |
| YAML | Yet Another Markup Language |
| XML | Extensible Markup Language |
| CNN | Convolutional Neural Network |
| R-CNN | Region-based Convolutional Neural Network |
| RPN | Region Proposal Network |
| FPN | Feature Pyramid Network |
| CLI | Command Line Interface |
| CSP | Cross Stage Partial |
| COCO | Common Objects in Context |
| MAP | Mean Average Precision |