Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Dental caries detection from bitewing X-ray images

Lukáš Kunt

Supervisor: prof. Dr. Ing. Jan Kybic
Field of study: Open informatics
Subfield: Artificial inteligence
May 2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Kunt**	Jméno: **Lukáš**	Osobní číslo: **478073**
Fakulta/ústav:	**Fakulta elektrotechnická**		
Zadávající katedra/ústav:	**Katedra počítačů**		
Studijní program:	**Otevřená informatika**		
Specializace:	**Umělá inteligence**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Detekce zubních kazů z rentgenových snímků

Název diplomové práce anglicky:

Dental caries detection from bitewing X-ray images

Pokyny pro vypracování:

1) Familiarize yourself with state-of-the-art methods in object detection and computer-assisted dental caries detection.
2) Based on the studied literature, propose a solution to dental caries detection from X-ray images based on deep learning.
3) Implement a program that automatically detects caries from bitewing X-ray images.

Seznam doporučené literatury:

[1] Srivastava et al: Detection of Tooth caries in Bitewing Radiographs using Deep .Learning. NIPS 2017 workshop on Machine Learning for Health.
[2] Jae-Hong Lee et al: Detection and diagnosis of dental caries using a deep learning-based convolutional neural network algorithm. Journal of Dentistry 2018
[3] Moran et al: Classification of Approximal Caries in Bitewing Radiographs Using Convolutional Neural Networks. Sensors 2021
[4] Yusuf Bayraktar, Enes Ayan: Diagnosis of interproximal caries lesions with deep convolutional neural network in digital bitewing radiographs. Clinical Oral Investigation, 2021.

Jméno a pracoviště vedoucí(ho) diplomové práce:

prof. Dr. Ing. Jan Kybic algoritmy pro biomedicínské zobrazování FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **23.02.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **19.02.2024**

prof. Dr. Ing. Jan Kybic podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.

Datum převzetí zadání	Podpis studenta

Acknowledgements

Firstly, I would like to express gratitude to prof. Jan Kybic for supervision of this thesis and his willingness to help at any time.

Secondly, I would like to thank MDDr. Tichý was always eager to help, and without his dedication to creating the dataset, we would not have achieved the results we did.

Last but not least, I would like to thank my family for their support during my studies.

I would like to further emphasize my gratitude to my girlfriend Anna for her undying moral and encouragement throughout the creation of this thesis.

Declaration

Abstract

Dental caries is the most prevalent disease globally, with more than 3.5 billion people affected. The treatment of dental caries imposes a burden on health care in every country financially and timewise. Detection of the disease in its early stages can mitigate the impact on the cost of treatment and improve the patient's prognosis.

Bitewing X-ray imaging is the second most used method for dental caries detection after the visual-tactile method. Aproximal and an-early stage carious lesion can be easily overlooked by the visual-tactile exam, making the bitewing X-ray imaging very beneficial for early detection and a chance for recovery without the need for further dental treatment.

This Master's thesis addresses the problem of dental caries detection from bitewing images using convolutional neural networks. First, a dataset of 3889 bitewing images with 7257 annotated dental caries was created for the purpose of this thesis. We trained multiple architectures for object detection and compared their performance using it. In the end, we used an ensemble of models to obtain the best-performing model.

We have created a solution that can detect dental caries with a precision of 0.751 and a recall of 0.7. Furthermore, a second model for segmentation of dental restoration was created, achieving an IOU score of 0.676.

Keywords: dental caries detection, convolutional neural networks, dental restorations segmentation, ensemble, bitewing, X-ray image

Supervisor: prof. Dr. Ing. Jan Kybic
Biomedical imaging algorithms, FEE

Abstrakt

Zubní kaz je jedním z nejrozšířenějších onemocnění na světě postihující více než 3.5 miliard lidí. Léčba je náročná jak finančně, tak časově a zatěžuje zdravotnický systém ve všech zemích světa. Včasná detekce zubního kazu umožňuje tuto zátěž snížit a zlepšit pacientovu prognózu.

Po detekci pohledem spojené s použitím zubařské sondy jsou bitewingové rentgenové snímky druhou nejvíce využívanou metodou pro diagnostiku zubního kazu. Časné a aproximální kariézní léze nejsou prvně zmíněnou metodou vždy spolehlivě diagnostikovány, což dává bitewingovým RTG snímkům značnou výhodu a šanci pro dřívější diagnostiku spojenou s možností zhojení léze bez nutnosti dalšího lékařského zásahu.

Tato diplomová práce se zabývá problémem detekce zubního kazu z bitewingových RTG snímků za použití konvolučních neuronových sítí. Pro účely této práce byl nejprve vytvořen dataset skládající se z 3889 bitewingových RTG snímků se 7257 anotovanými zubními kazy. Za jeho použití jsme natrénovali několik architektur pro detekci objektů a porovnali jejich výsledky. Nakonec jsme využili spojení modelů pro získání modelu s nejlepšími výsledky.

Vytvořili jsme řešení, které umožňuje detekci kazů s přesností 0.751 a citlivostí 0.7. Navíc byl vytvořen i druhý model pro segmentaci zubních výplní, který dosáhl IOU 0.676.

Klíčová slova: detekce zubních kazů, konvoluční neuronové sítě, segmentace zubních výplní, ensemble, bitewing, rentgenový snímek

Překlad názvu: Detekce zubních kazů z rentgenových snímků

Contents

Figures

Tables

Chapter 1

Introduction

As of 2017, dental caries was the most prevalent disease globally[20][21], with more than 3.5 billion affected people. Despite the advancement of technology in the medical field, the prevalence has not decreased, hence imposing a burden on health care in every country. In the US, more than six percent of total health care expenditures were targeted toward dental care in 2016[22].
Machine learning and especially neural networks have improved remarkably over the last decade, surpassing human-level performance in the ImageNet classification task in 2015[23]. Since then, deep learning models' error rates on the ImageNet dataset have become four times smaller[24]. This significant improvement in deep learning led to its wast adoption across many fields, including medical imaging. Deep-learning models exceeded human-level performance on specific tasks, such as breast cancer detection[25] or arrhythmia detection[26].
This thesis aims to develop a deep-learning-based model that will be able to detect dental caries in bitewing X-ray images. The position of every carious lesion is denoted by a minimal bounding box around the lesion. This model aims to give dentists an opportunity to cross-check their decision regarding the presence of carious lesions in the X-ray image. Furthermore, detecting caries' position directly from the image benefits dentists from saving information about dental caries in a digital form without their intervention. Having the data in a digital format could help dentists communicate the problem to a patient by overlaying the position of dental caries over an X-ray image or provide them an option to save that information for monitoring lesion progression over time. Last but not least, software like this would be helpful in education, where dentistry students would be able to train their ability to recognize dental caries without the need for a tutor.

The structure of this thesis is as follows: In Chapter 2, the medical background is introduced, describing human dentition and dental caries. Chapter 3 presents the fundamentals of the techniques used in this work. In Chapter 4, related work in automatic caries detection is discussed. Chapter 5 describes the dataset that was created for this thesis. In Chapter 6, we introduce the reader and the structure of the object detection framework that we created and used for caries detection. In Chapter 7, we propose a solution to the detection of dental caries. The results achieved by the proposed methods are presented in Chapter 8. Chapter 9 discusses the results and methods that we used to obtain them. In Chapter 10, we summarise the work and suggest future improvements.

Chapter 2

Medical background

2.1 Human teeth

Human dentition is composed of two sets of teeth - primary and permanent. The primary, also called deciduous, consists of 20 teeth and begins to erupt at six months of age. This dentition is completely replaced at the approximate age of 13 years by a permanent set of teeth, including 32 teeth. These can be divided into four classes based on function and form. Namely, those classes are:

Incisors

A total of 8 incisors teeth are found in primary and permanent dentition. They are located at the oral cavity entrance, and their primary purpose is to cut and shear food. They are essential for a smile's esthetics and play a vital role in phonetics.

Canines

A total number of four canines are located at the corners of dental arches, dividing them into a frontal and a lateral part. They have a triangular shape with a single cusp tip on the incisal edge. The structure is associated with their ability to seize, pierce, tear and cut food. Along with the incisors, they are essential for esthetics.

Premolars

Premolars are teeth found only in permanent dentition, being the successional teeth of all primary eight molars. Premolars share functional characteristics of canines and molars - they both seize and grind food thanks to their anatomy.

Molars

Human dentition contains 12 permanent molars with no deciduous predecessors. Their leading role is crushing and grinding food to dimensions appropriate for swallowing. Broad occlusal surfaces make them capable of this task. Molars are prone to dental caries due to deep grooves that run across the occlusal surface of

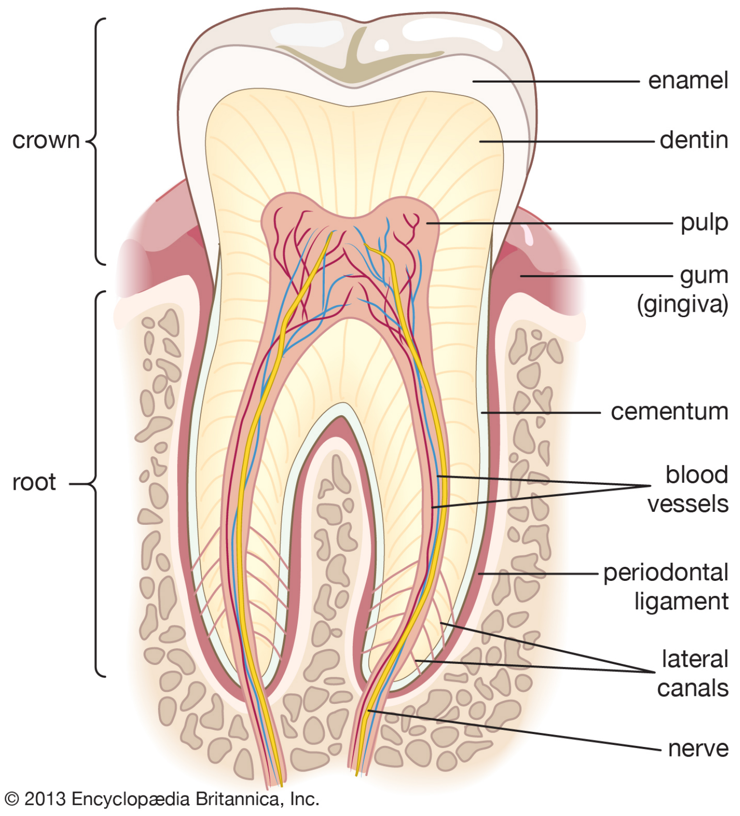


Figure 2.1: Structure of teeth

the teeth and a vast area of contact between adjacent molars. These places are difficult to clean, resulting in a space where bacterias tend to accumulate.

2.1.1 Structure of teeth

Teeth are composed of three structures: Enamel, pulp-dentin complex, and cementum. A picture of teeth structure is depicted in Figure 2.1. The superficial layer covering the anatomic crown of a tooth consists of a highly mineralized crystalline structure called the enamel. More than 90% of the volume is taken up by minerals (hydroxyapatite), making enamel the hardest substance of teeth and even the human body. Its thickness varies from one class of tooth to another, but it ranges from 2 to 3mm on average. Enamel is produced in the process of amelogenesis by cells occurring only in the development stage, meaning that it cannot regenerate. The biggest threat to enamel are acidic conditions, which can cause its demineralization. Enamel has the ability to remineralize, but if the cause is not removed, the enamel is irreversibly damaged, and a cavity is formed.

Pulp-Dentin complex

Pulp and dentin are two specialized connective tissues. However, some sources consider them a single tissue forming a complex [27]. The dental pulp is located in the pulp chamber of the tooth, and it serves four functions: formative, nutritive, sensory, and reparative. The pulp is circumscribed by dentin formed by specific cells in the process of dentinogenesis. Their cell bodies are found in the pulp chamber, but their cytoplasmic cell processes, located in dentinal tubules, extend into the mineralized dentin. Thanks to those processes, dentin is considered to be a living tissue. Its function is to provide the ability to regenerate and react to pathological stimuli, such as blocking the advancement of carious lesions by precipitating minerals in the affected area. Dentin forms the most significant portion of the tooth. In the coronal part, it is covered by the enamel, and on the root of the tooth overlayed by cementum. There are different types of dentin.

- **Primary dentin** forms the outer and most prominent layer of dentin closest to the enamel. It is produced in the development stage of the tooth.
- **Secondary dentin** is formed after the root development is completed.
- **Tertiary (reactive) dentin** production is encouraged as a response to pathological stimuli, such as injury or caries. It is produced at the pulp-dentin interface in order to protect the pulp.
- **Transparent dentin** is characterized by the presence of mineral precipitates in dentinal tubules as a result of injury or aging.

Cementum

Cementum covers the roots of teeth. Its structure consists of approximately 50 % of anorganic material, 50 % of organic matter, and water, making it slightly softer than dentin and far more delicate than enamel. Together with gingiva, periodontal ligaments, and the alveolar bone, cementum forms periodontium, ensuring that the tooth is attached to the bone. Cementum possesses the ability to repair itself to a limited degree.

2.2 Dental caries

2.2.1 Cause

Dental caries is an infectious disease characterized by the demineralization of hard dental tissues. The leading cause is dental plaque (also called a biofilm). Plaque is composed of bacteria, their by-products, and saliva, and it has the ability to adhere to the tooth structure. Some bacteria in the plaque metabolize refined dietary carbohydrates and produce organic acid by-products. If present in the biofilm for an extended period of time, those acids can lower the pH in the biofilm to below a critical threshold (5.5 for enamel, 6.2 for dentin)[27]. Low pH drives phosphate and calcium from the tooth into the biofilm in an attempt to reach an equilibrium. This loss of minerals in a tooth is called demineralization and, if not stopped, can lead to a caries lesion. However, this process can be controlled and eventually reverted if the pH returns to neutral and the relative concentration of soluble calcium and phosphate in the biofilm is higher than in the tooth. The cycle of demineralization and remineralization occurs multiple times a day and is modulated by many highly individual and tooth-specific factors.

2.2.2 Epidemiology

Untreated dental caries in permanent teeth is the most prevalent medical condition [20]. In 2010, around 35% of the global population was affected. The most considerable prevalence was observed around the age of 25. The sex of a person was not a significant factor in the statistics. No noticeable change in prevalence occurred between 1990 and 2010 [20] [28], which means that the technological improvement in dentistry did not affect the prevalence. Kassebaum et al. suggest that 42 new cases of tooth decay in primary and permanent teeth will develop annually from observing 100 people. This imposes a burden on health care systems. According to Huang et al., [22] in the United States alone, the cost of dental care in 2016 was 0.1 trillion $ out of total health care expenditure of 1.62 trillion $.

2.2.3 Diagnosis

Visual-tactile diagnosis is the primary way to inspect teeth and detect caries. Dentists use a mouth mirror and sharp probe to perform the examination. It is indispensable to dry teeth since the difference in the refractive index between sound and carious enamel is higher when water is removed from the tissue. This increases the chance of spotting a carious lesion before it has an opportunity to progress and cavitate the tooth. The second most used method clinicians use to complement the visual examination is a dental X-ray. In dentistry, two main types of X-ray imaging are taken during the examination: intraoral (the X-ray film is located inside the mouth) and extraoral (the X-ray film is outside the mouth). The intraoral images are the most commonly taken ones. This category includes bitewing and periapical X-rays, each featuring different aspects of the teeth. Extraoral imaging is mainly used to detect dental problems in the jaw and skull area. The most common one to be used is a panoramic radiograph [29].
Less common diagnostic measures are:

- Laser light-induced fluorescence
- Digital imaging fiber-optic transillumination
- Electrical conductance and impedance measurement

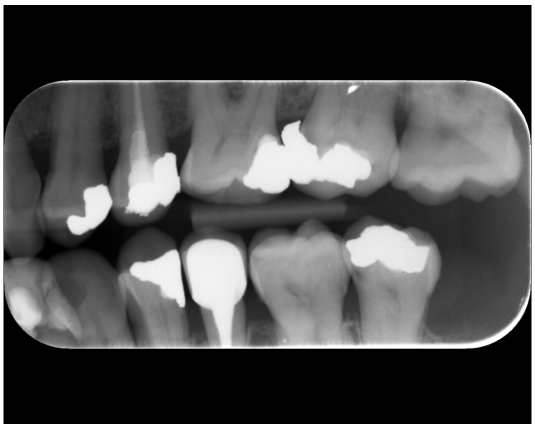


Figure 2.2: Bitewing X-ray image

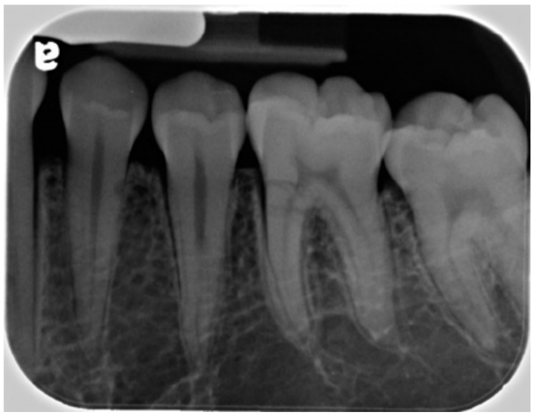


Figure 2.3: Periapical X-ray image, source [1]

Bitewing X-ray

The bitewing radiograph is an image that depicts the crowns of upper and lower teeth on the left or right side, as seen in Figure 2.2. It gives a clear sight of the interproximal surfaces allowing good caries detection in this area. Interproximal caries are challenging to diagnose by the visual-tactile method; thus, using the bitewing X-ray can lead to an early diagnosis and a chance for the enamel to remineralize. Also, bitewing X-rays portray the alveolar crest, where the dentist may notice any bone thickness changes due to periodontal disease. Unlike the other intraoral method, it does not show the entire length of the teeth. This type of dental X-ray is the most commonly taken for preventive purposes [29].

Classification of dental caries from bitewing X-ray

Dental caries are classified on multiple bases. The common ones are the depth of the lesion or lesion activity.
A frequently used classification scheme was proposed by Pitts & Fyffe in 1988 [27], including a total of 4 categories, three for cavitated lesions and one for non-cavitated lesions. They described this classification for oclusal surfaces. For bitewing X-ray images, the same classes are distinguished, only applied for the approximal surfaces of teeth.

- **D0** Surface sound. A healthy tooth with no evidence of either treated or untreated caries.
- **D1** Initial Caries. No detectable loss of tooth substance. Radiolucency is present in the outer half of the enamel.
- **D2** Enamel caries. Demonstrable loss of tooth substance. Radiolucency is visible in the inner half of the enamel and goes up to the enamel-dentin border. No evidence that cavitation has penetrated through the enamel layer into the dentin.

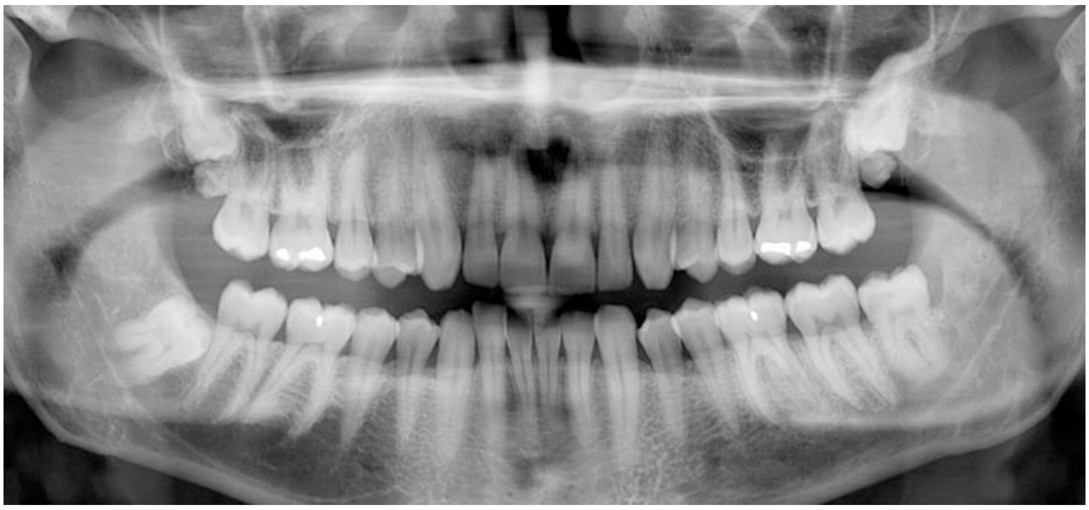


Figure 2.4: Panoramic X-ray image, source [2]

- **D3** Caries of dentin. Radiolucency reaches the dentine, breaking the enamel-dentine border. However, there is not a significant spread in the dentin.
- **D4** Pulpal involvement.Deep cavity forms with probable involvement of the pulp.

Periapical X-ray

Periapical X-ray portrays the tooth from the crown to where the root attaches to the jaw; hence, the whole tooth length is visible. As illustrated in Figure 2.3, it only shows the upper or lower teeth in one part of the jaw. Periapical X-ray detects any abnormalities in the root and any periapical lesions.

Panoramic X-ray

This extraoral dental image shows the entire mouth area, including the upper and lower jaw and adjacent structures. It depicts the full dentition, including teeth that have not erupted yet. Impacted teeth, i.e. wisdom teeth as seen in Figure 2.4, can be identified as well. Panoramic X-ray is often used before major procedures or to diagnose jaw tumors, cysts, fractures, or sinusitis. Nevertheless, it is not usually taken to diagnose dental caries [30].

Digital imaging fiber-optic transillumination (DIFOTI)

DIFOTI is different from the previously mentioned types of dental imaging. It works with infrared fiber-optic light and not an X-ray, unlike the others. A lesion's optical properties differ from those in healthy dental tissue, making it appear darker. DIFOTI enables the detection of fissure/occlusal caries, interproximal caries, and fractures and cracks in the tooth. It is a noninvasive method since it does not expose the patient to ionizing radiation [31].

2.2.4 Treatment

Treatment is suggested based on the progression of the lesion and the patient's risk profile. In some cases, only instructions to increase oral hygiene together with fluoride toothpaste are enough to stop the progression and lead to remineralization of the enamel. The dentist can suggest an application of a sealant to prevent further progression of the lesion. If this treatment is perceived as insufficient or if the carious lesion is already cavitated, restoring the tooth is required. This consists of removing all dental decay and filling the cavity with restorative material such as dental composite or amalgam [27][29].

Chapter 3

Theoretical background

3.1 Computer vision tasks

This section provides a brief overview of standard computer vision tasks.

3.1.1 Classification

Let's say we have an image $\mathbf{x}$. In a classification task, our goal is to assign one of n possible classes to the image:

$$\hat{y} = f_{\theta}\left(\mathbf{x}\right), \tag{3.1}$$

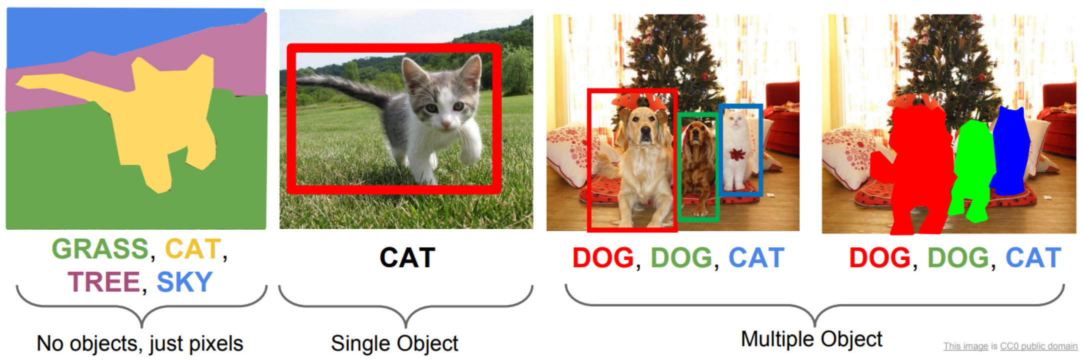
where f is a mapping, sometimes called a model, and θ represents model parameters if it holds that $\hat{y} = y$, where y is a true class of the image $\mathbf{x}$, the classification is considered to be correct. It is possible to output $\mathbf{p} \in \mathbb{R}^{n}$ instead of $\hat{y}$, where $p_i \in \mathbf{p}$ is probability of $i = y$, modeled by f_{θ}.

3.1.2 Semantic segmentation

For an input image $x \in \mathbb{R}^{n \times m}$, the goal is to output $\hat{\mathbf{Y}} \in \mathbb{R}^{n \times m}$, where $\hat{y_{i,j}}$ is the predicted class of pixel i, j in image $\mathbf{x}$. Similarly to the classification problem, we can output matrix $\mathbf{P} \in \mathbb{R}^{n \times m \times c}$, where $p_{i,j,c}$ is the probability of $pixel_{i,j}$ belonging to class c. A sample of semantic segmentation output can be seen in Figure 3.1.

3.1.3 Object detection

In object detection, the goal is to locate and recognize objects of interest in image $\mathbf{x}$. A rectangle and a category represent a ground truth object. Model predicts $\mathbf{Y} \in \mathbb{R}^{n \times 6}$ values for each image. Each row of $\mathbf{Y}$ consists of four numbers, which describe a rectangle, the category of the object inside the rectangle, and a number in the range from 0 to 1 called the confidence. In literature, we can see the term score instead of confidence. Nevertheless, the meaning remains the same: Certainty of the network regarding the particular prediction described by the bounding box and category. Please note that the confidence of predictions does not



.

Figure 3.1: From the left: semantic segmentation, object localization, object detection, instance segmentation

sum to one. In other words, we are not talking about probabilities since multiple detections per image can correspond to the ground truth.

3.1.4 Instance segmentation

Instance segmentation is similar to semantic segmentation, with the alteration saying that two objects of the same category would have different ground truth values. If we have $\mathcal{O}_1, \mathcal{O}_2$, where $\mathcal{O}_i \subset \mathbf{x}$ are pixels of object i in image $\mathbf{x}$. Then

$$o_{1_i} \neq o_{2_i} \quad \text{for} \quad o_{1_i} \in \mathcal{O}_1, o_{2_i} \in \mathcal{O}_2; \forall i. \tag{3.2}$$

3.2 Data format in object detection

As described in Section 3.1.3, the position of an object is denoted by a bounding box. The four parameters used to describe a bounding box can be selected in multiple ways. This ambiguity led to a disjoint notation. The most widespread are as follows.

3.2.1 PASCAL VOC

The format was introduced together with the PASCAL VOC dataset, the most popular dataset for object detection algorithm benchmarking in 2010. The bounding box is described by points $p_1(x, y), p_2(x, y)$ located in the top-left and bottom right corner. The coordinates range from 0 to image width/height in pixels. All the annotations are stored in a single XML file [32, 33].

3.2.2 COCO

COCO data format is represented by a single JSON file containing all bounding boxes of the dataset. The boxes are described by the top-left corner point $p(x, y)$ and the width and height of the box. The coordinates and box dimensions are again in the range 0 to image dimensions. In MS COCO, the annotation can be accompanied by a piece of additional information to solve the task as an instance segmentation problem.

3.2.3 YOLO

This format was introduced together with the first YOLO architecture [34], and this annotation style is still persistent whenever working with the YOLO-family neural networks. In this format, the annotations are divided into multiple TXT files and each of them contains annotations for a single image. The bounding box is described similarly as in the COCO dataset, but the coordinates are normalized to be in the 0 to 1 range. The advantage of this approach is that the annotations do not have to be modified when image dimensions are scaled [34, 33].

3.3 Metrics

3.3.1 Intersection over union (IOU)

Intersection over union, also known as the Jaccard index, is defined as demonstrated: Let B_{gt} and B_p be a ground truth and a predicted bounding box. The Jaccard index J is calculated as

$$IOU = J(B_p, B_{gt}) = \frac{area(B_p \bigcap B_{gt})}{area(B_p \bigcup B_{gt})}. \tag{3.3}$$

From the Equation 3.3, we can observe that the lowest value of IOU is 0. This means there is no overlap and the maximal value is 1, indicating a perfect match. We use a predefined threshold value of IOU to decide if the predicted bounding box matches the ground truth. Usually, we choose this threshold to be 0.5 or above.

IOU can be defined for the semantic segmentation task with two classes (e.g. background and target class). Let $\hat{\mathbb{Y}} \in \mathbb{R}^{m \times n}$ be the mask of the predicted values, where $\hat{y}_{i,j} = 1$ if the model predicts, that pixel i, j belongs to target class. The IOU is defined as:

$$IOU = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \hat{y}_{i,j} \wedge y_{i,j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} \hat{y}_{i,j} \vee y_{i,j}}, \tag{3.4}$$

where $y_{i,j} \in \mathbf{Y} \in \mathbb{R}^{m\ timesn}$ is the ground truth value for pixel i, j.

3.3.2 Precision and recall

Precision

When speaking about object detection, we say that a prediction is a true positive (TP) if the IOU value is greater than the predefined threshold τ. If otherwise, the

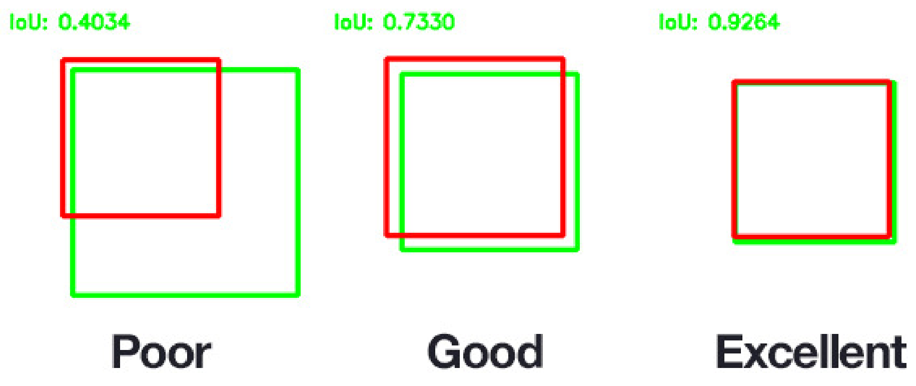


Figure 3.2: Examples of IOUs for different overlaps between GT and predicted box, source. [3]

prediction is treated as a false positive (FP). Let's assume there are N predictions of our model, from which S are correct. Precision is defined as

$$Precision(\tau, \gamma) = \frac{TP(\tau, \gamma)}{TP(\tau, \gamma) + FP(\tau, \gamma)}, \tag{3.5}$$

where γ is the confidence threshold, meaning we discard all predictions with confidence smaller than γ. Note that for fixed value of τ are $FP(\gamma)$ and $TP(\gamma)$ decreasing functions of γ [33].

Recall

If there is a ground truth bounding box for which there are no given detection values of γ and τ, we say it is a false negative (FN). If we consider a dataset with G ground-truths and N predictions of which S is correct, where $(S \leq G)$, the recall is expressed as:

$$Recall(\tau, \gamma) = \frac{TP(\tau, \gamma)}{TP(\tau, \gamma) + FN(\tau, \gamma)}. \tag{3.6}$$

Since the value of $FN(\gamma)$ increases with the growing value of γ, we see that recall is the decreasing function of γ.

F1 score

The value of the F1 score is computed as the harmonic mean of precision and recall.

$$F1(\tau, \gamma) = \frac{2 \cdot Recall(\tau, \gamma) \cdot Precision(\tau, \gamma)}{Recall(\tau, \gamma) + Precision(\tau, \gamma)}. \tag{3.7}$$

Precision-recall curve (PR curve)

From Subsections 3.3.2 and 3.3.2, we were able to observe that precision mainly grows as we increase the confidence threshold γ, while at the same time recall decreases. The precision-recall curve captures the relation between precision and recall. An example of the curve is illustrated as the blue line in Figure 3.3. In other words, we can say that the precision-recall curve is a mapping

$$\gamma \rightarrow Precision(\gamma) \times Recall(\gamma), \tag{3.8}$$

where γ ranges from 1 to 0.

Mean average precision (mAP)

To calculate mAP we first need to get the PR-curve and then interpolate the precision values. Suppose that we have K different confidence values γ among model predictions, which are ordered as

$$\gamma(k),\ k = 1, 2, ..., K, \text{ such that } \gamma(i) > \gamma(j)\ for\ i > j. \tag{3.9}$$

The interpolated precision-recall curve is then defined as

$$Precision_{interp}(R) = \max_{k|Recall(\gamma(k)) \geq R} \{Precision(\gamma(k))\}, \tag{3.10}$$

where R is a real value contained in interval [0,1] [33]. The interpolated precision-recall curve is pictured in Figure 3.3 in red color. Now, we can compute the average precision (AP) as the area under the interpolated PR curve. In practice, there are two different ways to approach the Reimann integral. They differ in the number of samples used to compute the integral and are called N-point and all-point interpolation. The N-point, specifically 101 point interpolation, is used in the MS COCO competition. On the other hand, the all-point interpolation is nowadays used in PASCAL VOC challenges [4, 4].

Since the AP is calculated per class, the mean average precision is defined as the average in all categories.

In Subsections 3.3.2 and 3.3.2, we stated that precision and recall depend on a predefined IOU threshold τ to consider prediction as true positive. This dependency makes the value of MAP vary over different values of τ. The threshold value used for computation of the mAP is usually denoted in the metrics name, such as $AP@.5$ in the case of MS COCO. [1] The standalone AP without any numerical values attached to it usually refers to the official COCO metric. The official COCO metric is in its explicit form written as $AP@[.5 : 0.05 : 0.95]$ and is computed as the average of MAP values for ten different τ values, ranging from 0.5 to 0.95.

The letters S, M, and L in the subscript, such as AP_S denote that the metrics are calculated for a subset of ground truth predictions only. Taking into the consideration only bounding boxes with area $\leq 32^2$ pixels, $32^2 \leq$ area $\leq 96^2$ pixels and area $> 96^2$ pixels

[1] Note that even though the mean average precision is computed, the AP shortcut, which stands for average precision, is used.

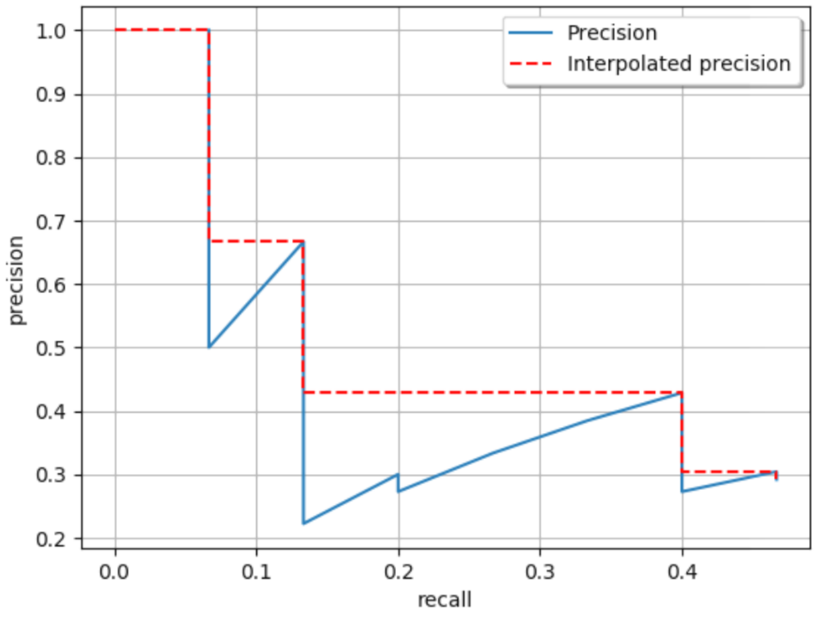


Figure 3.3: Standard and interpolated precision-recall curve, source [4]

3.3.3 Mean average recall in MS-COCO (mAR)

PyCOCOtools, the official metrics for MS-COCO benchmark [35], compute the average recall (AR) by the following approach: Predictions are sorted according to their confidence in a decreasing order. We take n boxes with the highest confidence values and evaluate their recall by Equation 3.6 for a predefined IOU threshold θ. We use a similar notation as in the case of AP, where $AR@.X_{na}$ denotes the average recall computed for IOU threhsold X, where we consider n most confident predictions. By $a \in \{S, M, L, \text{all}\}$ we denote size of the rectangles, for which AR is computed. If we omit some of those, the following default values are used $a = \text{all}, n = 100, X = [.5 : 0.05 : 0.95]$.

Cross-Entropy loss

Let $\mathbf{y} \in \mathbb{R}^n, y_i \in \{0, 1\}$ be a vector of ground truth classes and $\hat{\mathbf{y}} \in \mathbb{R}^n$ be a vector of model preditions, where $\hat{y}_i \in [0, 1]$ is the predicted probability, that i_{th} element belongs to class 1. The cross-entropy loss is computed as follows [36]:

$$L_{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(p(\hat{y}_i)) + (1 - y_i) \log(1 - \hat{y}_i) \tag{3.11}$$

Soft Dice Loss

Let's consider $\mathbf{y}$ and $\hat{\mathbf{y}}$ as in section 3.3.3, Soft Dice Loss is then computed as:

$$SDL(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{\sum_{i=1}^{N} 2y_i\hat{y}_i}{\sum_{i=1}^{N} y_i + \sum_{i=1}^{N} \hat{y}_i} \tag{3.12}$$

Dice Loss is computed in the same way, we only threshold values of $\hat{\mathbf{y}}$ prior to computation of the loss [37, 36].

3.4 Optimization

In deep learning, a defined loss function that should be minimized usually does not have an analytical solution, or the solution cannot be evaluated for computational reasons. Therefore, the iterative numerical optimization approach is used, where we compute the gradient of the loss function with respect to the parameters of the optimized model. Those are updated by changing their values in the negative direction of the computed gradient.

3.4.1 Optimizers

The most simple optimizer is Stochastic Gradient Descent (SGD), which in each step updates the weights by stepping in the opposite direction of the gradient The learning rate affects the length of the step.

Many advanced optimizers that increase the speed of convergence are available. Commonly used are SGD with moentum, Adam and AdamW.

3.4.2 Weight decay

We can add the L_2 of the model weights to the loss function. This term is called weight decay and should decrease the discrepancy between performance on training and testing part of the dataset.

3.4.3 Learning rate schedulers

Learning rate is considered to be one of the most, if not the most important parameter, in deep learning. It is usually beneficial to change the learning rate during the course of training. This can be done manually or automated by an algorithm that increases or decreases the learning rate based on the set of predefined rules. This algorithm is called the learning rate scheduler.

ReduceLROnPlateau

We couple the scheduler with a model metric, and when the improvement of the metric stalls for a predefined period, the learning rate decreases. The scheduler is not heavily reliant on the setting of its hyperparameters, making it a go-to starting choice for most developers.

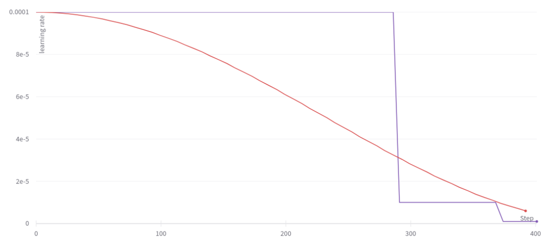


Figure 3.4: Learning rate schedulers: Cosine annealing is red, Reduce learning rate on plateau has purple color

Cosine annealing

Cosine annealing changes the value of learning rate according to the equation 3.13, where T is half-period of the cosine.

$$lr(t) = lr_{min} + \frac{1}{2}\left(lr_{max} - lr_{min}\right)\left(1 + \cos\left(\frac{t}{T}\pi\right)\right) \tag{3.13}$$

It is commonly used with two different settings. Either we set T to estimated length of the training. The learning rate than decreases throghout the training, as can be seen in Figure 3.4, or we select small value of T and the learning rate oscilates in predefined boundaries multiple times throughout the training. This should help the optimizer to overcome saddle points.

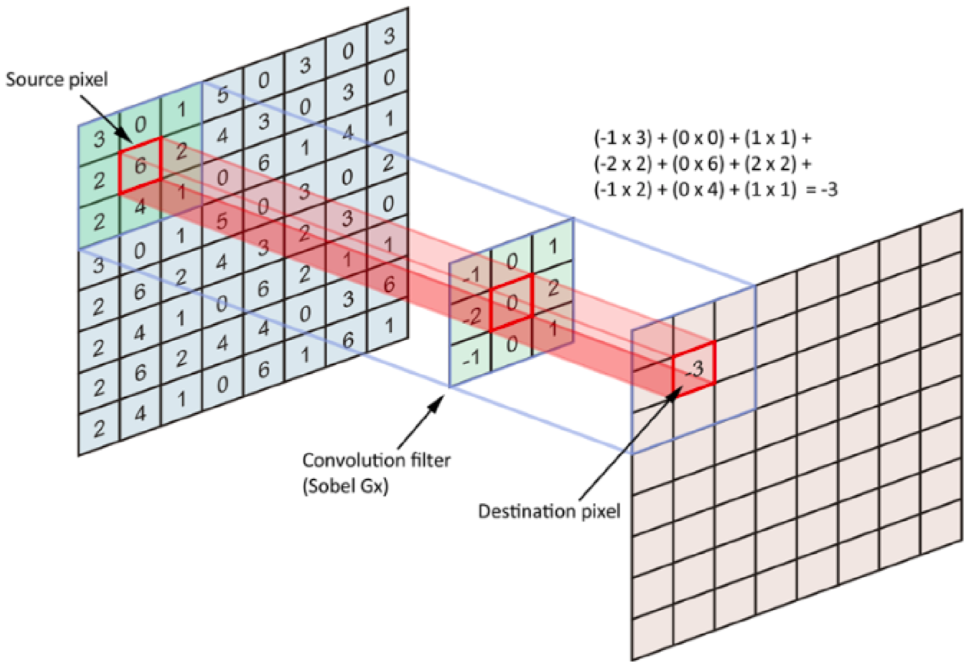
3.5 Artificial neural network (ANN)

The mechanisms of the human brain inspire artificial neural networks. Human neuron cells are in ANN replaced by artificial neurons, which are defined as:

$$y = f\left(\boldsymbol{w}^T\boldsymbol{x} + b\right). \tag{3.14}$$

Where $\boldsymbol{x}$ is a vector of inputs, $\boldsymbol{w}$ stands for weights and b is bias. Symbol f denotes an activation function f : $\mathbb{R} \to \mathbb{R}$. The artificial neuron proposed by Frank Rosenblatt in the perceptron algorithm worked with a step function [38], but nowadays, different functions such as ReLu, sigmoid or tanh are used. The output of the neuron y is called activation of the neuron.

Neurons are usually structured into layers. The connection between layers depends upon the architectural choice. First ANNs used fully connected layers, meaning that the input into a neuron in layer n was composed of all activations from layer $n - 1$. Fully connected neural networks are nowadays sparsely used in computer vision. Convolutional neural networks (CNNs) or vision transformers are used instead. In the case of CNNs we limit neurons' receptive field to the local neighborhood only; this decreases the computation complexity and includes our prior knowledge of pixel neighborhood in the input image. Having the same weights for the whole input makes the network invariant to shifts in the input.


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

3.5.1 Convolutional layer

A convolutional layer consists of C_{out} neurons, each having C_{in}, H, W receptive field. Those neurons are called kernels with width W height H and several input channels C_{in}. In each layer, we convolve[2] the input X with the kernel W, the output Y is defined by:

$$y_{o,i,j} = \sum_{c_{in}} \sum_{\Delta i} \sum_{\Delta j} x_{c,i+\Delta i,j+\Delta j} w_{o,c,\Delta i,\Delta j} \tag{3.15}$$

Nowadays, modifications of convolutional layers are proposed, such as dilated convolution, grouped convolution, or depth-wise separable convolutions are used. However, the fundamentals remain the same: Filter sliding over the input produces an output.

3.5.2 Activation functions

A non-linear activation function usually follows the output of the convolutional layer. Many activation functions are at our disposal, but the most commonly used is ReLU and its derivatives, such as SERLU, SELU, ELU, Swish, and Leaky ReLU. Values of those functions are depicted in Figure 3.5.

3.5.3 Normalization layers

Normalization layers make the training of ANNs faster and more stable. It has been shown, that normalization-layers decrease the generalization gap ,while increasing

[2]Even though we usually refer to convolution, in practice, cross-correlation is used instead. Terms cross-correlation and convolution are used interchangeably.

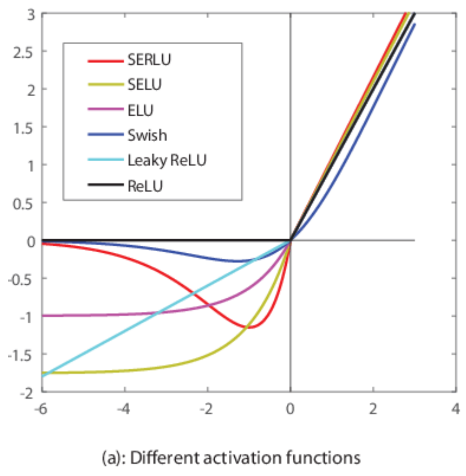


Figure 3.5: Graphs of ReLU based activation functions, source [5]

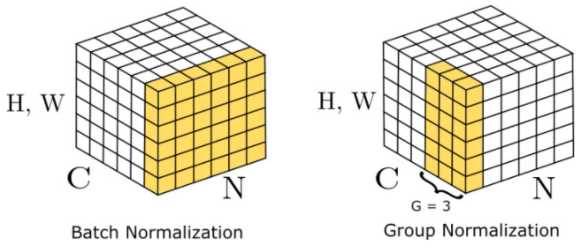


Figure 3.6: Batch and group normalization layers with denoted axes, across, which the normalization statistics is computed

the convergence speed [39, 40]. The normalization layers differ in spatial axis, across which, the normalization statistics is computed, this is illustrated in Figure

Batch-normalization

The most normalization layer is batch normalization, which computes the output of the layer y_i as:

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}^2} + \epsilon}}; \qquad y_i \leftarrow \gamma \hat{x}_i + \beta \tag{3.16}$$

where γ and β are learnable parameters, $\mathcal{B}$ denotes, that this value is computed over a mini-batch.

Group-normalization

Wu et al. [40] proposed a normalization method, where the statistics are computed over groups of channels. We et al. showed that group normalization outperforms batch normalization when both layers are used with batch sizes smaller than eight.

The disadvantage of group normalization is the introduction of a new group size G hyper-parameter, which needs to be tuned to obtain the results claimed by the authors [40].

3.6 Transformer architecture

Transformer architecture debuted in computer vision in 2021 and has achieved outstanding results, beating state-of-the-art models in multiple benchmarks across all computer vision tasks. As of May 2022, the best-performing models in the main computer vision benchmarks are based on transformer architecture. We think of the following benchmarks to be the main ones in computer vision: ImageNet benchmark (classification task), COCO (object detection), ADE20K (semantic segmentation).

The transformer architecture was proposed already in 2017 for the task of natural language processing (NLP). We will briefly introduce transformer architecture for the NLP task since it is crucial for understanding transformers for computer vision.

Transformers in NLP

Transformer architecture was introduced in the paper Attention is all you need [41] for NLP. NLP is a task where input is a sequence of words of length n and output is a sequence of m words, where n and m usually differ. The sequence of words is converted into a sequence of vectors. There are multiple options for how to embed words into the vector. Commonly used is TD-IDF or Word2Vec[42]. Positional encoding is added to those vectors are then the encoder block processes it. The novel key component is the self-attention module, where for the input sequence of vector values V, keys K and queries Q are computed. We output values and keys from the encoder, and from the decoder's self-attention module, we output queries. We then take keys and values from the encoder and queries from the decoder and input them into another attention block:

$$Attention\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3.17}$$

where d_k is dimension of keys. More details can be seen in Figure 3.7 or in [41].

3.6.1 Transformers in computer vision

The first transformer-based model was the Vision Transformer (ViT) which is capable of image classification only. It is composed of multiple encoder blocks stacked on top of each other; those blocks are the same as those used by the transformer for the NLP task. On the top encoder block is attached a multi-layer perceptron (MLP) head, which outputs values for each class. Those can be converted into corresponding probabilities by a softmax layer. The input into ViT are 16×16 image patches linearly projected into vectors; the whole architecture is shown in Figure

3.7 General architecture for object detection

Even though there is a wide variety of architectures for object detection, the core principles remain the same. The model is composed of three main parts: backbone, neck, and head, as depicted in Figure 3.11. Each of those blocks can usually be swapped for a different one, fulfilling the same purpose. This gives us great flexibility and allows us to try different combinations of those blocks.

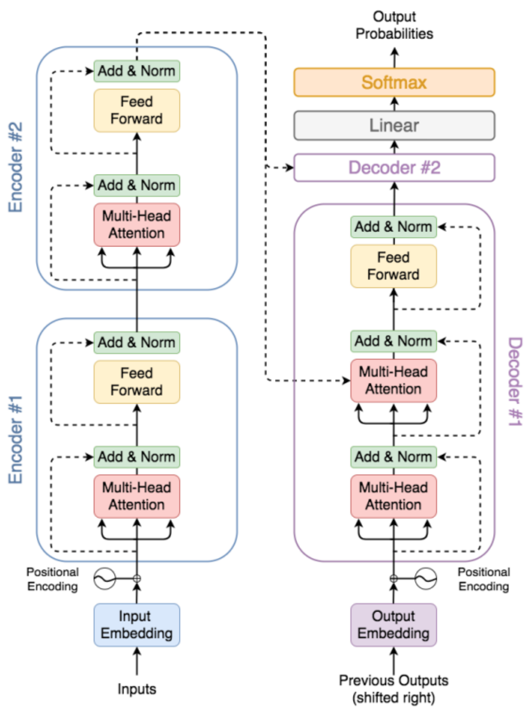


Figure 3.7: Architecture of transformer with two encoders and two decoders, source [6]

Backbone

The purpose of backbones is to transform the input image into feature maps. For this purpose, we use classification models with the classification head removed. Most parameters of object detection models are usually part of the backbone. The extraction of useful feature maps is vital for other blocks to perform well. The most common backbones are models from the ResNet family.

Neck

The neck is responsible for the merging of features from the backbone. This is not a straightforward task since we usually use features from different backbone layers. This allows us to get semantically strong features from deeper layers and more detailed information from earlier ones. Common neck architectures are feature pyramid network or PANet [43].

Head

The head is responsible for predicting the position of boxes and their classification. It uses the features extracted by the backbone and merged by the neck. Based on the approach to box prediction, we differ them into Dense prediction heads (YOLO, RetinaNet) or Sparse prediction heads(Faster R-CNN) [44].

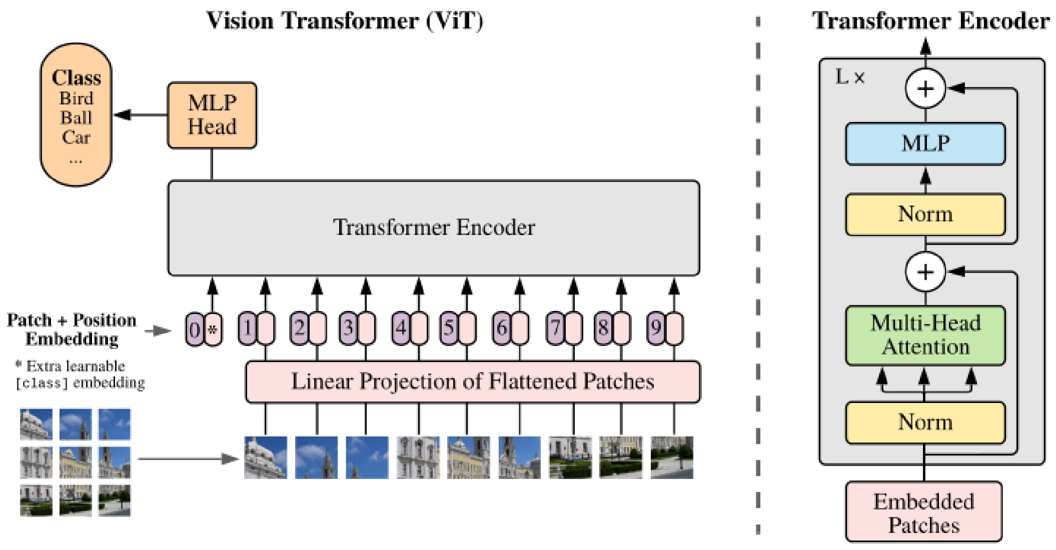


Figure 3.8: Architecture of ViT, source [7]

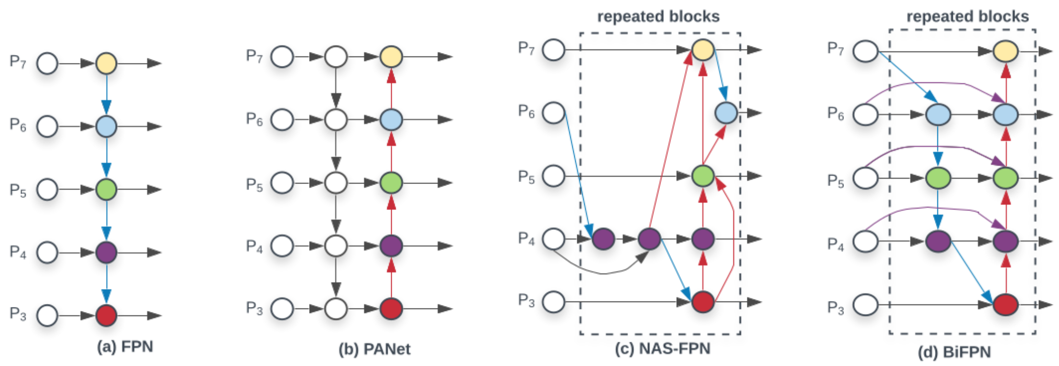


Figure 3.9: Architecture of different necks for feature fusion, source [8]

3.8 Backbone models

This section will introduce multiple architectures of neural networks, which were used as a backbone throughout our work.

3.8.1 ResNet

ResNet architecture was introduced by He et al. [45] and proposed a novel element of deep-learning architectures - an identity shortcut connection sometimes called a skip-connection. Let x be the input into a block composed of multiple convolutional layers with activation functions in between[3]; we will call this block a mapping $\mathcal{F}$.

[3]Addition of batch-normalization, or other layers is possible

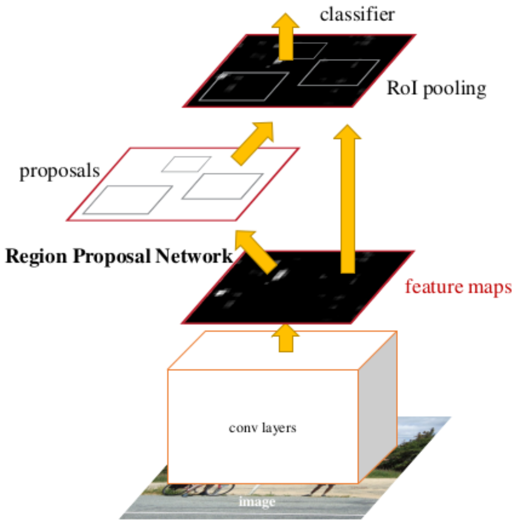


Figure 3.10: The schema of two-stage detection process of Faster R-CNN

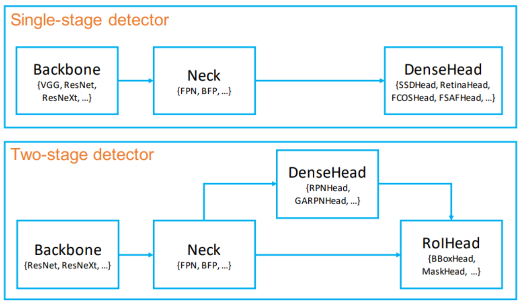


Figure 3.11: General architecture for object detection, source [9]

The output of the residual block $\mathcal{H}$, derived from $\mathcal{F}$ is defined as:

$$\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}. \tag{3.18}$$

The reasoning behind the residual block is to make it easier to learn the identity mapping if desired. This has other benefits, especially the improvement of the gradient flow during back-propagation, making it easier to optimize such blocks. This ease of optimization can be seen by inspecting the loss function landscape of a model with and without skip-connections in Figure 3.12. Final ResNet architecture is composed of multiple residual blocks stacked one after another; the models vary in the number of layers used; this is denoted in the name with a number such as ResNet50 or ResNet101.

3.8.2 EfficientNet

When scaling the model's size, we can increase: Number of layers (depth), the number of filters in each layer (width), or the width and height of feature maps (resolution). It has been a common practice to change only one of them. Tan and Le [46] did a multi-objective neural network search, where they tried to maximize objective function O defined as:

$$O = ACC(m) \times [FLOPS(m)/T]^{w}, \tag{3.19}$$

where $ACC(m)$ and $FLOPS(m)$ are accuracy and floating-point operations (FLOPS) of model m, T is the target number of FLOPS, and w is a hyper-parameter controlling the trade-off between accuracy and number of FLOPS of the final model. This search resulted in the EfficientNet-B0 baseline model, which can be scaled to obtain a more extensive network called B1-B7.

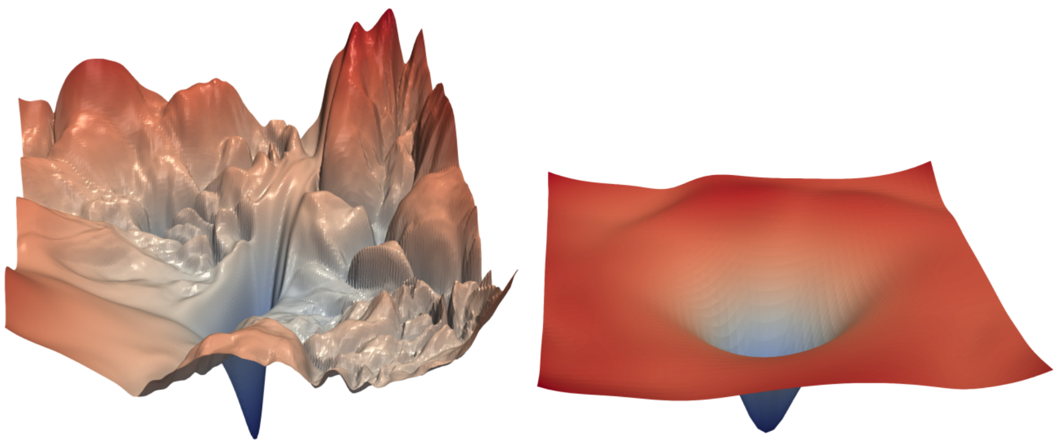


Figure 3.12: Comparison of loss landscapes, source [10]

3.8.3 Swin transformer

Swin transformer architecture overcomes the limitations of ViT, which is working with 16×16 image patches only. This is insufficient for segmentation and object detection tasks, where dense predictions are needed. Swin transformers are in the first layer working with 4×4 patches. Since the computation complexity of the self-attention layer grows quadratically with the number of input tokens, the authors overcome this by using neighbor patches only. Attention is thus computed with respect to tokens in the non-overlapping local window. As depicted in Figure3.14, this local window for computing self-attention is shifted after every encoder layer. This shift introduces cross-window connections, which increase the modeling capacity of the model. After a particular number of encoder layers, neighbor patches are merged, which reduces the number of patches while increasing their size by a predefined factor. This mimics the behavior of CNNs, where we start with big, semantically weak feature maps and gradually decrease their dimensions while increasing their number. Having this kind of feature map allows using swin transformer as a general backbone for any task. [11]

3.9 Detection models

3.9.1 Faster R-CNN

Faster RCNN (Region-Based Neural Network) architecture is a two-stage detector. In the first stage, Region Proposal Network (RPN) finds regions of interest (ROI)and proposes bounding boxes corresponding to those regions. This is done by sliding a small neural network over the output of the backbone. In the second stage, features corresponding to positions of ROIs are extracted from the backbone and processed by a classification network, which decides if the region corresponds to a

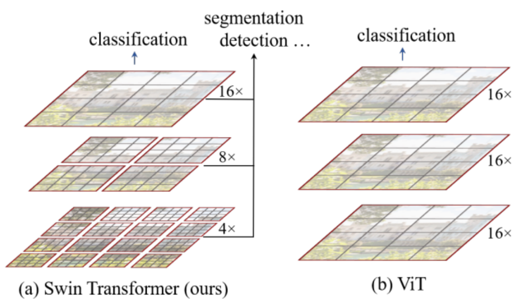


Figure 3.13: Hierachical structure of Swin Transformer compared with ViT, source [11]

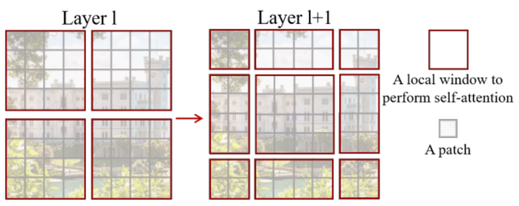


Figure 3.14: Shift of local window for computation of self-attention, source [11]

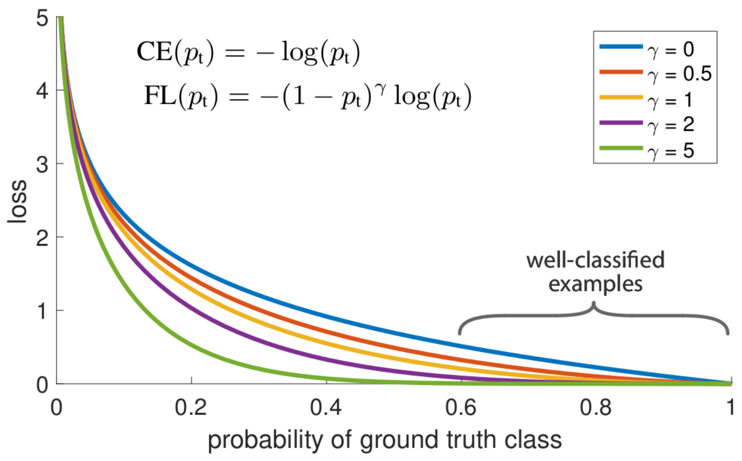


Figure 3.15: Focal Loss, source [12]

background or is one of the target classes. The schema of the architecture is in Figure 3.10

3.9.2 RetinaNet

The biggest contribution of RetinaNet is the introduction of focal loss [12], see Figure 3.15. It helps to mitigate to the problem if class imbalance by changing the formula Cross-Entropy loss. The penalization of well-classified samples decreases, increasing the importance of correct classification of hard-to-classify examples.

3.9.3 EfficientDet

EfficientDet tries to achieve a similar goal as EfficientNet: Propose a computationally effective architecture for object detection that would be scalable. Since an efficient backbone architecture was already proposed [46], they focus mainly on feature fusion from multiple layers. Based on the study of FPN, PANet, and NAS-FPN, a Bidirectional feature pyramid network (BiFPN) was proposed as the

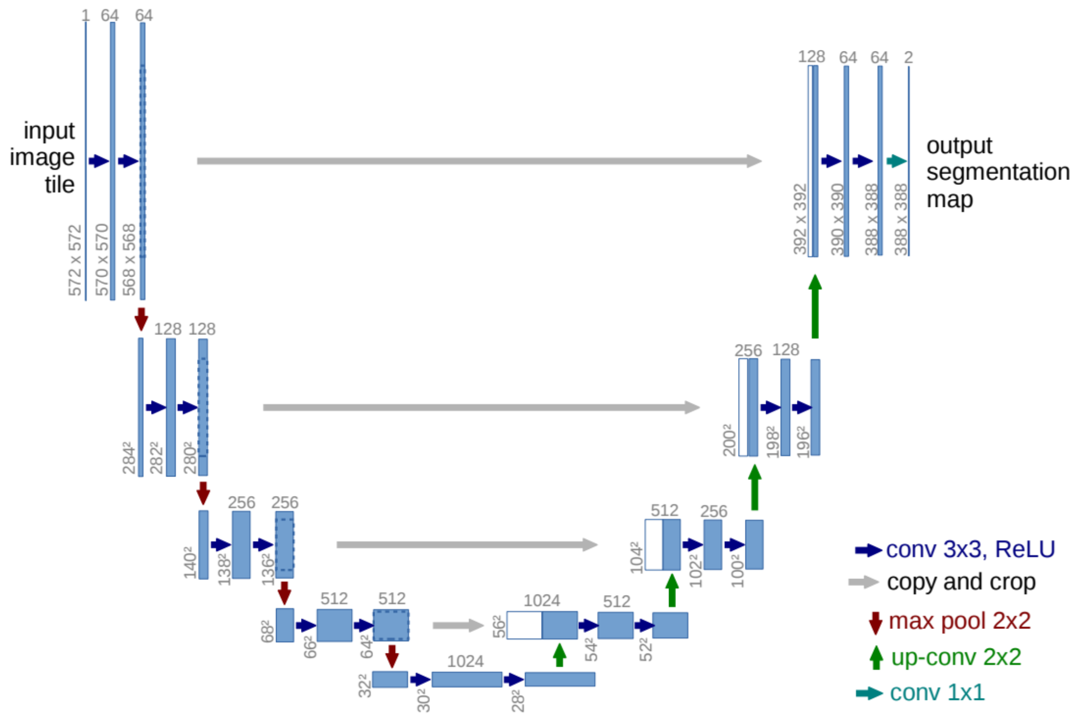


Figure 3.16: Architecture of U-Net model, source [13]

most computationally effective neck architecture [8]; it consists of multiple BiFPN blocks stacked on top of each other, see 3.9. The count of those blocks is dependent on the size of the used backbone.

3.9.4 Models for image segmentation

U-Net

U-Net is an architecture for semantic segmentation, with an encoder-decoder structure as shown in Figure 3.16. The decoder extracts feature maps from the input image with an increasing semantical strength throughout the layers. In the middle of the network is a so-called bottle-neck layer with the strongest semantical information about the image but lacks information about high-resolution details of the input image. Hence when the decoder decodes the information from the bottle neck, it is combined with information from the shallower layer, which contains information about image details required to obtain a precise dense prediction.
The decoder proposed by Rennenberger et al. [13] can be replaced by a general-purpose backbone, as demonstrated by Baheti et al. [47], who used EfficientNet as the backbone.

```
Input  : $\mathcal{B} = \{b_1, .., b_N\}$, $\mathcal{S} = \{s_1, .., s_N\}$, $N_t$
         $\mathcal{B}$ is the list of initial detection boxes
         $\mathcal{S}$ contains corresponding detection scores
         $N_t$ is the NMS threshold
begin
    $\mathcal{D} \leftarrow \{\}$
    while $\mathcal{B} \neq empty$ do
        $m \leftarrow \text{argmax } \mathcal{S}$
        $\mathcal{M} \leftarrow b_m$
        $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
        for $b_i$ in $\mathcal{B}$ do
            if $iou(\mathcal{M}, b_i) \geq N_t$ then
                $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
            end                                         NMS

            $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$          Soft-NMS
        end
    end
    return $\mathcal{D}, \mathcal{S}$
end
```

Figure 3.17: Pseud code of NMS and soft-NMS, source [14]

3.10 Model ensembling in object detection

Let say we have M different models, each of them predicting $\mathcal{B}_i = \{b_1, ..., b_N\}$ bounding boxes and $\mathcal{S}_i = \{s_1, ..., s_N\}$ confidence values for a given image corresponding to a single class. We merge predictions of all models together. It is possible to use weights $W = \{w_1, ..., w_M\}$ to express our prior belief in the given model. Set of all boxes $\mathcal{B}$ and confidence scores $\mathcal{S}$ is thus obtained by:

$$\mathcal{B} = \bigcup_{i=1}^{M} B_i \quad , \mathcal{S} = \bigcup_{i=1}^{M} \frac{S_i w_i}{F}, \tag{3.20}$$

where F is an optional normalization constant. It's only purpose is to ensure that confidence score will be less than 1 after the ensembling. Commonly used value for F is $\frac{1}{M}\sum_{i=1}^{M} w_i$. After obtaining sets $\mathcal{B}$ and $\mathcal{S}$ we post-process them by one of the following algorithms: Non-maximal suppression, soft non-maximal suppression, non-maximum weighted suppression or weighted boxes fusion.

Non-maximal supression (NMS)

In non-maximal suppression, we first sort all boxes $\mathcal{B}$ by their confidence $\mathcal{S}$ in descending order. We go thru the sorted set $\mathcal{B}$ and check if any other box b in $\mathcal{B}$ has an overlap greater than the predefined threshold N_t. In that case, we remove box b from $\mathcal{B}$. More details regarding the NMS algorithm are in the pseudocode, which is in Figure 3.17.

Soft non-maximal suppression S-NMS

Soft non-maximal supression extends the NMS algorithm by a possibility to keep overlapping the predictions if their confidences are high. Instead of removing boxes with an overlap, we decrease their confidence value by the follwing Gaussian penalty function:

$$s_i = s_i e^{-\frac{\mathrm{iou}(\mathcal{M},b_i)^2}{\sigma}}, \forall b_i \notin \mathcal{D} \tag{3.21}$$

where $\mathcal{M}$ is the currently processed bounding box, and $\mathcal{D}$ is the set of already processed boxes. After processing all boxes, those with $s_i < T$ are removed, where T stands for a confidence cut-off threshold [14].

Non-maximum weighted suppression (NMW)

Non-maximum weighted suppression does not remove boxes in case of an overlap, but merges them togehter by following formula:

$$\mathcal{M} = \frac{\sum_{i=1}^{n} \omega_i \times b_i}{\sum_{i=1}^{n} \omega_i} \tag{3.22}$$

$$\omega_i = s_i \times \mathrm{iou}\left(b_i, b_{argmax_i s_i}\right) \tag{3.23}$$

where $\mathcal{M}$ is the merged bounding box, for which no confidence value is computed [48, 49].

Weighted boxes fusion (WBF)

Weighted boxes fusion combines boxes similarly to NMW. The main difference is the iterative approach to the fusion, outputting confidence for the merged box, and awareness of several models, which contributed to the prediction. The steps of WBF are as follows [49]:

1. Sort $\mathcal{B}$ by $\mathcal{S}$ as in NMS.
2. Declare empty lists L and F that would be used to store boxes clusters and merged boxes, respectively
3. Iterate through $\boldsymbol{\mathcal{B}}$. If there is a box in F for which $\boldsymbol{IOU} > \mathbf{Threshold}$, add the box from $\boldsymbol{\mathcal{B}}$ to list L on the position corresponding to position of the matched box in F. If there is no match found, add it to the end of L.
4. Recalculate the box coordinates $\boldsymbol{\mathcal{M}}$ and confidence $\boldsymbol{c}$ in the list F on the position where we added the box to L by formulas3.24.
5. After processing all boxes from $\boldsymbol{\mathcal{B}}$ adjust confidence scores by a formula3.25, where $\boldsymbol{T}$ is the number of contributing boxes and $\boldsymbol{M}$ is the number of models used for ensembling.

$$\mathcal{M} = \frac{\sum_{i=1}^{n} c_i \times b_i}{\sum_{i=1}^{n} c_i}, \quad c = \frac{\sum_{i=1}^{T} c_i}{T} \tag{3.24}$$

$$c = c * \frac{T}{M} \tag{3.25}$$

Chapter 4
Related Work

This chapter will introduce relevant publications regarding dental caries detection, focusing mainly on detection from bitewing radiographs. Following that, we will briefly introduce methods for the segmentation of dental restorations.

4.1 Dental caries detection

Since 2017, more than ten publications have been released regarding automatic caries detection from images [50]. They differ in how they approach caries localization and the types of images they use. The following images have been used: Near-Infrared Transillumination images [51, 52], camera photographs [53], and X-ray images, which may be further classified into bitewing [54, 55, 56, 57, 16], panoramic [58], and periapical X-ray images [59].

All the related publications can be divided into three groups based on their approach to caries localization: Manual detection and classification, dental caries segmentation, and dental caries detection.

4.1.1 Manual detection and classification

This section introduces publications that approached caries detection in the following manner: First, they crop individual teeth from the X-ray image, using manual cropping or non-machine learning computer vision techniques. After the tooth is extracted from the image, it is labeled by a professional. A classifier is trained on those image patches to decide if it contains a carious lesion.

- First attempts to use a neural network for caries detection date back to 2008, when Kuang et al. [60] proposed an approach based on passing a patch from an image to a classifier, which then decided if the patch contains caries or healthy enamel. Even though the performance of the proposed neural network was surpassed by 6.72% by kernel SVM, it was still able to outperform an ordinary dentist by more than 5%. It was only 6% worse than an experienced individual.

- Moran et al.[54] used histogram equalization, Otsu’s thresholding, and morphological operations to extract individual teeth from bitewing images. After the teeth had been cropped from the image, the dataset was labeled by assigning one of three categories to each tooth. The categories were: Normal teeth, incipient lesions, and advanced lesions. Moran et al. processed a total of 112 radiographs this way, resulting in 480 teeth with corresponding annotations. They trained the ResNet and Inception model to perform the classification task, and the best-achieved accuracy was 73.3% [54].

- Mao at al. [57] made a similar preprocessing approach as Moran, only this time extracting unilateral tooth images instead of the whole tooth. A total of 3716 images of unilateral teeth were obtained. AlexNetsed was used for classification and reached a 90.3% accuracy.
- Lee et al. [59] published a similar approach, however, with periapical images. A dataset with 3000 images was created manually by cropping out teeth from the X-ray image, keeping only those without extensive dental restorations. Two teeth at once were also extracted from the radiograph in the same process. After obtaining this dataset, they trained GoogLeNet and Inception v3 architecture classifiers, reaching an accuracy of 89% for molars and 82% for images with both premolars and molars.

4.1.2 Dental caries segmentation

There are publications where the authors approached the task of caries localization as semantic segmentation. The advantage of this approach is the pixel precision of the lesion detection. On the other hand, creating a similar dataset is very time-consuming. An example of a dataset annotated in a pixel-wise manner is depicted in Figure 4.2 as well as predictions of a model proposed by Cantu [55].

- Cantu et al. [55] created a dataset of 3686 bitewing images. Three dentists drew a polygonal-shaped box over caries independently in each image. In the case of a unanimous decision, the annotation was kept in the dataset. Otherwise, the fourth dentist reviewed the annotation and decided if it should be kept or deleted. Cantu et al. used the U-Net model with EfficientNet B5 as a backbone. They then evaluated the model per pixel, and its performance was compared against seven dentists, outperforming their mean performance in every metric.
- Lian et al. [58] chose the same approach as Cantu but used panoramic images. In comparison with Cantu, following the segmentation, they cropped the region of interest around the segmented lesion and classified caries into one of four categories as described in Section 2.2.3. They achieved an IOU score of 0.785 on the segmentation task. In comparison, the best performing dentist achieved an IOU of 0.717. In the classification task, the model outperformed the average dentist's performance.
- Lee et al. [15] approached the problem uniquely. Their dataset, consisting of 304 bitewing radiographs, was densely annotated by polygons, denoting the

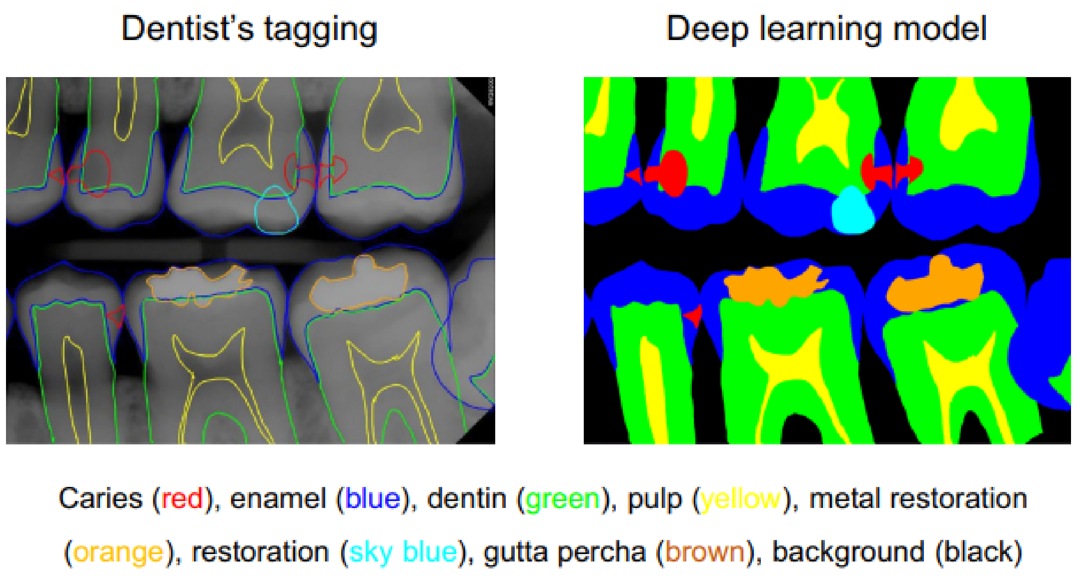

Dentist's tagging
Deep learning model
Caries (red), enamel (blue), dentin (green), pulp (yellow), metal restoration (orange), restoration (sky blue), gutta percha (brown), background (black)

Figure 4.1: Annotated bitewing radiograph and the same image post-processed, source [15]

position of dental caries, enamel, dentine, pulp, and gutta-percha restorations. The result of this annotation can be seen in Figure 4.1. They used two independent U-net models to predict the position of caries and remaining structures in the image. The output of both models was post-processed and merged. Even though the model achieved an F1 score of 0.641, which is low compared to other publications, predictions of the model helped dentists improve their sensitivity ratio by 7 - 10%.

4.1.3 Dental caries detection

- Srivastava et al. [16] trained a fully convolutional neural network with over 100 layers on a dataset containing more than 3000 bitewing radiographs. They denoted the position of tooth decay in a pixel-wise manner. Even though the model predicts output masks in a semantic segmentation fashion, the output is post-processed by fitting a minimal bounding rectangle around the prediction, as can be seen in Figure 4.3. After that, the model is evaluated by computing the IOU of the rectangle with the ground truth polygon. If the IOU is greater than 0.8, the detection is considered positive. Srivastava et al. claim that their model considerably outperforms each of the three dentists included in the study. Detailed results are in Table 4.1.

- The same author and Kumar [19] published another paper, where they changed the model to U-Net, which was trained on an extended dataset of 6000 bitewing X-ray images. The authors tested the hard example mining approach, but it led to a decrease in performance. Even though U-Net architecture usually

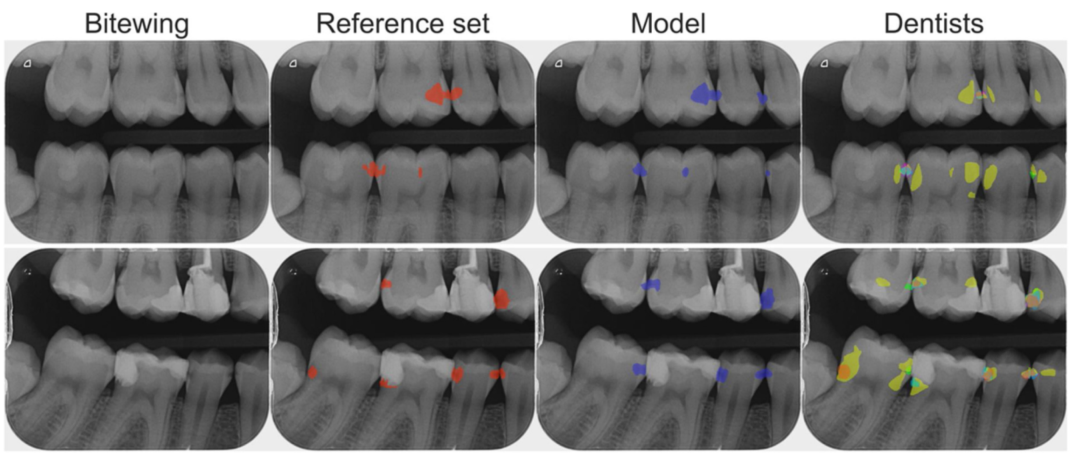


Figure 4.2: Sample of data and predictions of the model by

Metric	Model [16]	Model [19]	Dr. 1	Dr. 2	Dr. 3
Recall	0.805	0.70	0.477	0.433	0.344
Precision	0.615	0.53	0.63	0.815	0.891
F1-Score	0.70	0.614	0.54	0.56	0.50

Table 4.1: Results of models proposed in [16] and [19], compared with three dentists, modified

achieves better results on publicly accessible benchmarks [24, 61], and the size of the dataset increased twofold, the model's performance dropped by 15%, see Table 4.1. There is no information available about the evaluation protocol used by Kumar [19], nor about the IOU threshold needed to consider a prediction to be correct. This makes it hard to estimate the cause of the performance drop.

- Barakdar et al. [56] did both semantic segmentation and object detection with a dataset of 621 bitewing images available for both of those tasks. They claim to use U-net for segmentation and VGGNet for object detection. However, the paper does not mention how they modified the VGGNet architecture for object classification to perform an object detection task. The object detection results were evaluated against five professionals in dentistry with different years of experience. The model outperformed two dentists with two and three years of experience while being outperformed significantly by all three dentists with ten years of experience. The reported precision of the model is 0.78, recall=0.77 and F1 score of 0.78. No information about the overlap used to consider predictions to be correct is included in the paper. We assume it was set to be 0.5.

- Bayraktar et al. [62] solved only the object detection task on a dataset of 1000 bitewing images labeled by two experts with more than ten years of experience. With YOLOv3 architecture model, they achieved $AP@.5 = 0.872$.

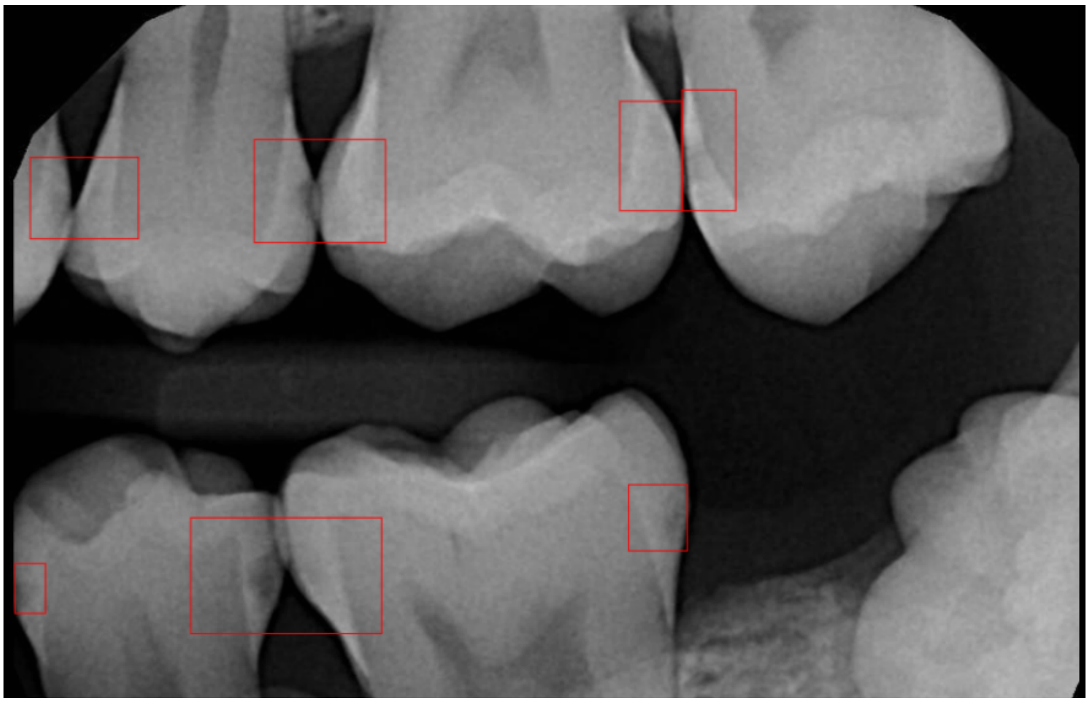


Figure 4.3: Predictions of the model proposed by Srivastava et al., source [16]

4.2 Dental restorations segmentation

To the author's knowledge, there are no available publications regarding the segmentation of dental restorations in bitewing radiographs. Therefore, we will introduce two methods that segment dental caries from panoramic X-ray images. Figures 4.4, **??** contain samples of images used for restoration segmentation. In addition, we will mention two publications where restoration detection was a minor part of the work.

- Mao et al. [57] classified dental segmentations in previously extracted image patches with unilateral teeth.
- Lee et al. [15] did not focus directly on the segmentation of restorations, yet it was one of the classes segmented out by their U-net architecture. There are no metrics available regarding the algorithm's performance on dental restorations.
- Abdalla-Aslan et al. [18] used methods of classical computer vision to segment out restorations in panoramic images. Their pipeline consisted of: Adaptive gaussian thresholding, morphological operations, and deleting regions in peripheral areas of the image. The final algorithm had the precision and sensitivity of 0.33 and 0.946, respectively. After successful detection, the restoration was classified as: dental implant, crown, amalgam filing, etc.

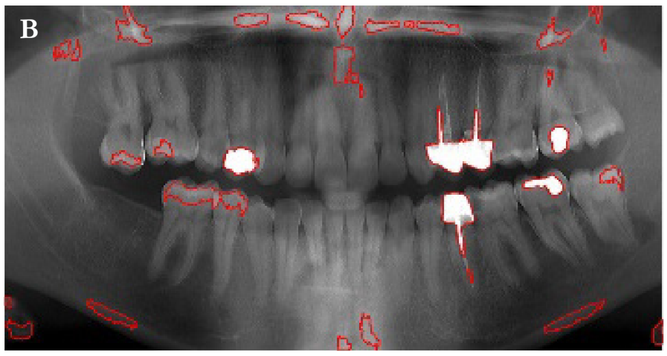


Figure 4.4: Results of segmentation algorithm proposed by Yeshua et al.[17]

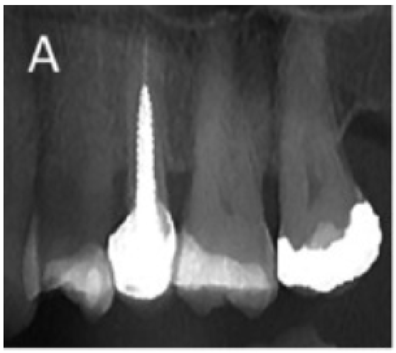


Figure 4.5: Cropped region from panoramatic image with multiple restorations, source [18]

- Yeshua et al. [17] solved the same problem as Abdalla-Aslan. Even the approach was more-less the same, except theirs achieved a precision of 0.568. They classified detected areas similarly to Abdalla-Aslan, having an extra category for false detections. After the removal of false detections, the precision was boosted to 0.98.

Chapter 5
Dataset

In total, MDDr. Tichý and his team created two datasets. One dataset was used to detect dental caries and the other for semantic segmentation of dental restorations. The majority of work was done on the first-mentioned set of data.

5.1 Dental caries

MDDr. Tichý and his team began working on the datasets in September of 2021, along with the beginning of work on this thesis. This led to an opportunity to discuss the format of the data. We decided to annotate every dental caries lesion with a minimal bounding box. The annotation process was conducted in the Computer Vision Annotation Tool (CVAT), running on the Faculty of biomedical informatics server. The web address is www.gdiag.fbmi.cvut.cz.

While denoting the position of the carious lesion, the annotator tried to be consistent with the following guidelines:

- Carious lesion is marked by a rectangle. The rectangle should contain the entire lesion while remaining as small as possible.
- When the lesion is on the proximal surface, and if both teeth are infected, draw a separate box for each.

Due to constant work on the dataset, we decided to use the same data as long as there was no major update that would lead to a release of an improved dataset. In total, we did six major releases. Let’s call these releases the stages of the dataset. This ensured that we were able to compare the performance of our models to each other in different stages of the dataset.

First stage

In the very first stage MDDr. Tichý instructed a group of dentistry students on how to approach the annotation to get as homogenous dataset as possible. They then annotated a couple of images under his supervision before continuing independently. Dental X-ray images were uploaded into CVAT and divided into

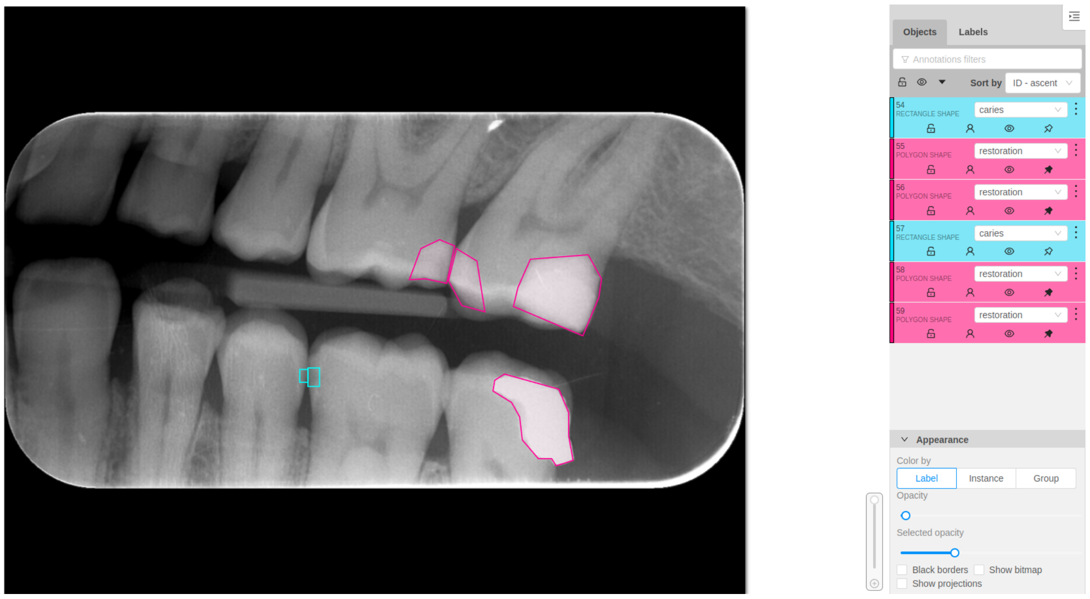


Figure 5.1: The environment of CVAT with annotated carious lesion and dental restorations

multiple projects, where each project contained between 400-800 images. This was essential due to technical limitations regarding exporting and uploading X-ray images from a dental database. We further split each project into jobs consisting of 100 images each and assigned them to a particular student. We had 1695 X-ray images at our disposal with 2416 dental caries annotated after the first stage was done. CVAT does not allow exporting and merging multiple tasks. Hence we exported each task separately in a COCO format. We uploaded all images and files with annotations to the CMP server. The server contains annotations combined in one annotation.json file, carrying the information about the dataset in COCO format and one folder with all the images. We checked the task for duplicates and non-reviewed radiographs and removed those, which resulted in 1626 images with 2399 decay annotations. Out of those, 946 images contained at least one cavity.

Second stage

After an inspection of the dataset created in the first stage, we observed inhomogeneity across the annotations. Some of the guidelines were violated, especially the one regarding caries on proximal surfaces. In addition, we observed multiple overlooked lesions. This led us to a reconsideration of our approach to labeling and MDDr. Tichý himself did all the annotation work from this moment further on. After the second stage, the dataset was extended to 2599 non-duplicate images containing 4328 annotations of tooth decay. During this stage, we did no corrections of previous errors.

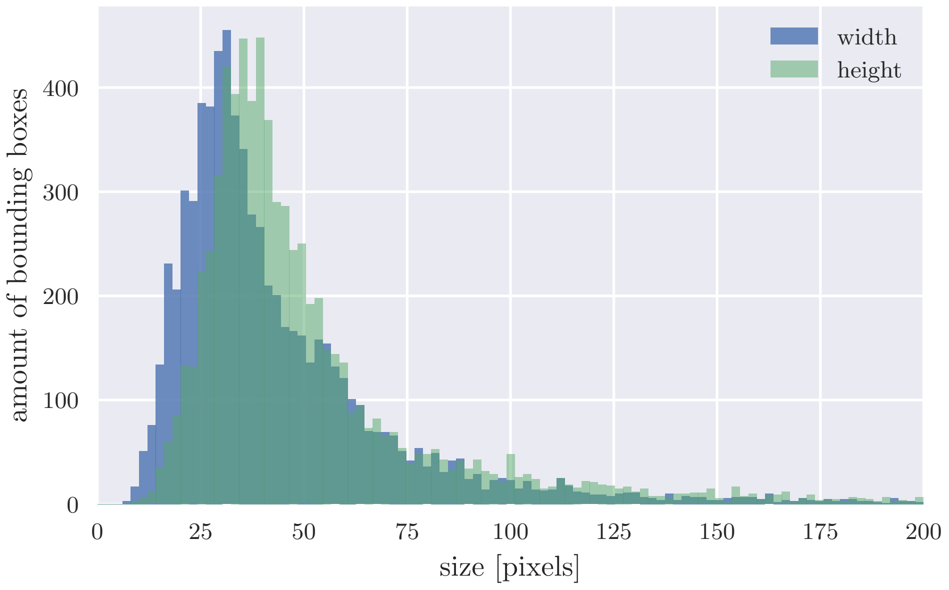


Figure 5.2: Histogram of bonding boxes dimensions in the dataset

Third stage

MDDr. Tichý reviewed all images annotated in the first stage, removing as well as adding an unspecified amount of annotations. In the end, the dataset consisted of 2599 images with 4575 annotations of dental caries.

Fourth stage

We uploaded another 1400 images onto the CVAT server. All the images were downloaded and uploaded to the CMP server. We used a YOLOv5 model[1] trained on the third stage dataset to generate predictions for each image. Confidence threshold maximizing F1 score on the validation dataset was used to filter out low confidence predictions of the model. We used Voxel Fiftyone tool to upload all 1400 images and their respective predictions to CVAT, where those images were split into two separate tasks. MDDr. Tichý reviewed all predictions and conducted adjustments to bounding boxes and their removal and addition. According to his personal statistics, there were roughly 200 predictions per 100 images. Around 20 predictions had to be added and removed to get the same quality annotations as in stage three. The speed was the upside of using model prediction as a starting point for the annotation process. The annotation was done in approximately half the time required to do the annotation without model predictions. In total, 3500 images were available subsequent to this stage. After removing corrupted images, we got 3489 X-rays with 6087 annotations.

[1]For further information about the performance of the model, see Table 8.3

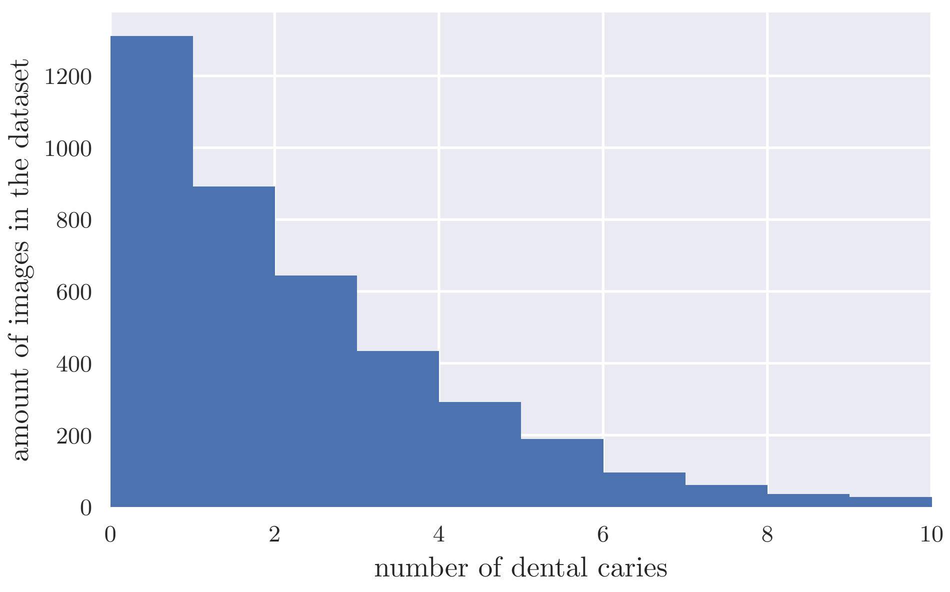


Figure 5.3: Histogram of the number of dental caries per image

	Width [px]	Height [px]
Image size	1068	795-847
Minimal box size	8	9
Maximal box size	384	315
Mean of box size	47.55	53.15
Standard deviation of box size	37.99	35.33

Table 5.1: Statistics of bounding boxes that denote position of carious lesions

Image augmentations

Fifth stage

In this stage, we finished the annotations of all 1400 images uploaded in stage four, resulting in 3989 X-rays with 7257 annotations. We plotted a histogram representing the distribution of the number of carries among images; it can be noticed in Figure 5.3.

Sixth stage

We evaluated the model's performance on the test, validation, and training part of the dataset. Although the model used for prediction achieved $AP@.5 = 0.72$, there were 1598 images with at least one false positive or false negative detection. We decided that doing a second round of dataset review would be more beneficial than further expansion of the size of the dataset. We focused only on erroneous images and uploaded those 1598 images with no less than a single error to the CVAT annotation tool for revision. During the time of writing this thesis, there is

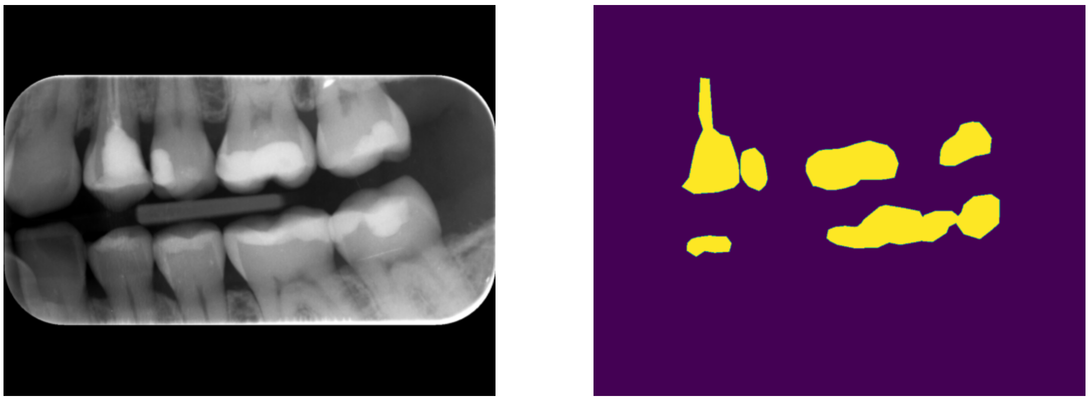


Figure 5.4: Bitewing X-ray image on the left, pixel mask of the X-ray on the right (dental restorations have yellow color)

still undergoing work on the uploaded images. Therefore, we were not able to use the sixth stage.

5.2 Dental restorations

This dataset consists of a subset of images used in the dental caries dataset. MDDr. Tichý's team annotated them in the CVAT tool by drawing a polygon around each dental restoration. The work was done by the same group of dentistry students as stage one of the caries dataset, and reviewed by a single fifth-year dentistry student. Evaluation of the whole dataset performed by a single person should ensure consistency among images. A total of 521 images were used to create this dataset, and an inspection revealed that 387 radiographs contained at least a single annotated restoration, and 134 had none. The dataset was exported from CVAT in COCO format and saved on the CMP server. When working with the data, we used a pixel mask instead of polygons to denote the position of dental restorations. A sample of the dataset with a pixel mask is featured in Figure 5.4.

In Figure 5.5, we notice how many percent of the X-ray image consists of restorations., which gives us an idea of how common dental restorations in our data are. In Figure 5.6, we see how the size of each restoration is distributed. We observe that most restorations are smaller than 2% of the image by inspecting this Figure.

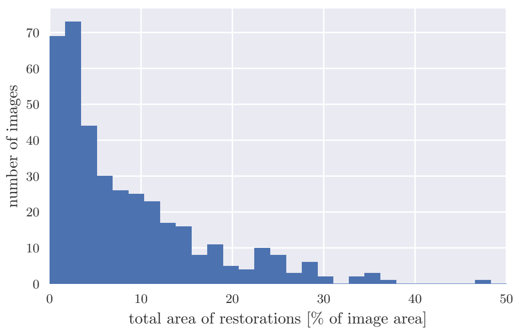


Figure 5.5: Histogram of restoration area in image, images without restorations omitted

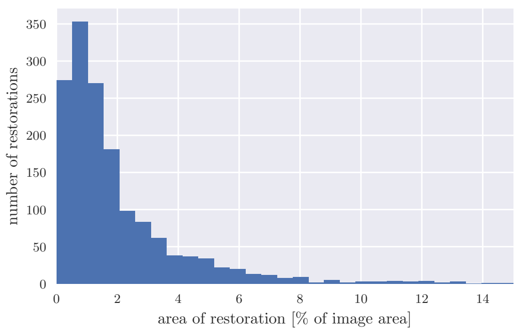


Figure 5.6: Histogram of areas of restorations, 10 largest omitted

Chapter 6

Project structure

6.1 Organization of the project

The object detection framework developed for this project was programmed in Python 3.8.12 [63]. We structured it into multiple independent modules, which can be swapped for their corresponding alternatives. This ensures maximal reuse of the written code and allows further extension of this framework. This modular approach was inspired by MetaAI research [64] and IceVision library [65]. The framework's modularity will enable us to use it for semantic segmentation or classification problems. The core of the project is the deep-learning framework PyTorch 1.11.0 [66], which is used to create and train neural networks. Although there are many open-source libraries with object detection frameworks, relying only on those libraries is far from optimal because the options to change the program's behavior are limited. We, therefore, decided to write our framework. It handles all tasks required for training a model: Loading the data, model definition, optimization of the model, tracking the progress of training, etc. Implementing all models from scratch would be ineffective; therefore, we support third-party models' usage. However, only the bare model is used. Thus, we can change everything except for the architecture of the model and its forward pass.

The project is divided into three folders: configurations, tests, and source (src). The first-mentioned contains YAML files that are further dispersed into dedicated folders based on the module they configure. At the root of the configuration folder are train.yaml and test.yaml files, which define how to compose individual modules to perform the target tasks. Hydra [67] handles the composition of configuration files. The user can override the default configuration from the command line or experiment.yaml file. The whole configuration pipeline can be seen in Figure 6.1, and the project's folder structure is in Figure 6.2.

6.1.1 Models

Models are implementations of one of the architectures described in section 3.9. We can either fully implement them, or we can rely on open-source libraries. In that case, we need to implement a function that transforms the data into the format

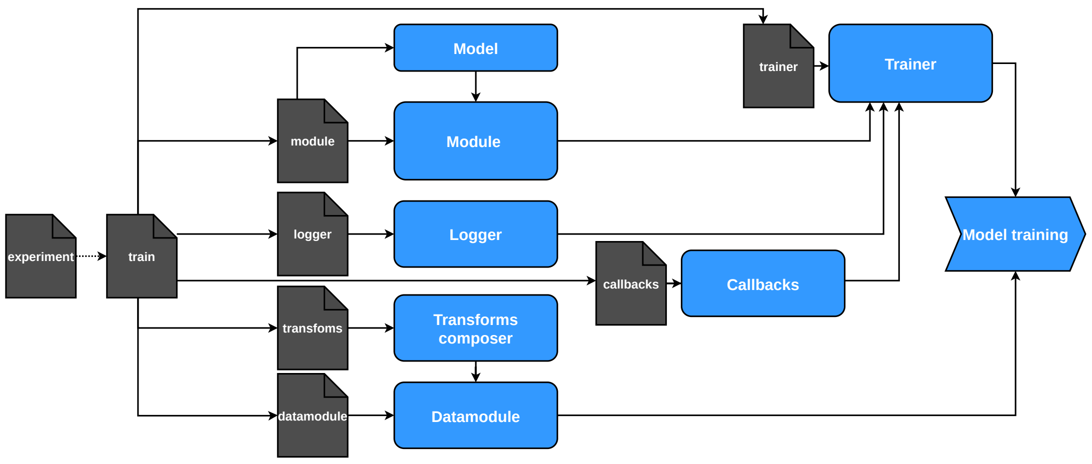


Figure 6.1: Structure of modules and their configuration files

required by the third-party model.

6.1.2 Modules

Modules are a wrapper around the model and they are based on Pytroch-Lightning modules [68]. In modules, we take care of the following:

- Training and validation loop
- Initialization of optimizer
- Initialization of learning rate schedulers
- Computing metrics
- Logging metrics to a predefined logger
- Transformations of outputs of models into the unified format

6.1.3 Transformations

Transformations are defined by their YAML configuration file. This file is passed to the transformation composer class, which creates training and validation transformations, which are then passed to a data module. We relied on the Albumentations library for individual image augmentations.

6.1.4 Data-Modules

Data modules are based on PyTorch-Lightning data modules. They consist of:

- Dataset, where we load the data from the hard drive into the memory and parse the file with saved annotations. After loading the data, predefined transformations are applied.

```
MT
├ configs - root folder with all configuration files
│ ├ callbacks
│ ├ datamodule
│ ├ experiment
│ ├ logger
│ ├ module
│ ├ trainer
│ ├ transforms
│ └ train.yaml
├ src - folder with all source files
│ ├ core - structures for data manipulation
│ ├ data - utility functions for dataloaders and datasets
│ ├ datamodules
│ ├ models
│ ├ modules
│ ├ notebooks - jupyter notebooks for auxilary tasks
│ ├ transforms - classes to compose transformations given by configuration file
│ └ utils - functions for logging, losses, data conversion
└ tests - contains multiple subfolders with unit tests for the program
```

Figure 6.2: Folder structure of the project

- Functions that transform the loaded data into the format required by the model we are about to train.
- Definitions of data loaders, which are responsible for merging the data into chunks, so-called batches.

6.1.5 Trainer

Trainer defines properties of the training, such as the number of GPUS used for the training or a maximal number of epochs that training is allowed to undergo before terminating.

6.1.6 Callbacks

Callbacks add capabilities to the training pipeline without changing any code. Typical ones that we used were: Definition of stopping criteria, adjusting policy for saving weights of the model, or saving images with their predictions during the training.

6.1.7 Logging

A logger is software capable of storing and visualizing logged values. In the beginning, we used Tensorboard TODO, but soon we switched to Weights and Biases [69], and used them throughout the rest of the thesis.

6.2 Additional open-source software

Throughout the project, we used the libraries mentioned above as well as the following:

- We used OpenCV [70] computer vision library during the segmentation of dental restorations for operations such as: Adaptive thresholding, morphological operations, etc.
- Computer vision library Kornia [71] was used to perform morphological operations on PyTorch tensors quickly.
- MMDetection [72] provides an implementation of multiple object detection models. We used their implementation of swin transformers, RetinaNet and Faster-RCNN
- During the computation of metrics, we used PyCOCOtools, an official [35] implementation of MS COCO metrics. It had to be significantly modified to provide us with the required capabilities.
- To visualize predictions, we used Voxel Fiftyone [73]. The program can be on a self-hosted server. We used this to share predictions of the model with MDDr. Tichý, who was thus able to assess those and decide if the dataset contains any erroneous annotations.
- For model ensembling, we used the methods implemented by the author of Weighted box fusion [49], and we further enhanced the capabilities of their methods.
- We used YOLOv5 models from the Ultralytics repository. [74].

Chapter 7
Methods

This chapter describes our proposed solution. It is divided into the following blocks.

Section 7.1 describes the baseline solution. We mainly describe the training protocol used in the remaining sections of the chapter.

Section 7.2 introduces a few changes to the training protocol previously described in Section 7.1.

Section 7.3 inspects how the behavior of trained models changes when we choose a differently sized backbone or use different weight decay.

Section 7.4 improves detection results by ensembling multiple models. Furthermore, we assess the importance of models used in the ensembling.

Section 7.5 proposes a deep learning and a non-deep learning approach to segmentation of dental restorations.

7.1 Caries detection - baseline model comparison

Firstly, we implemented and tested multiple object-detection architectures and compared them against each other on the currently available stage of the dataset. The training protocol used throughout the training of all models is described by the following:

7.1.1 Dataset

We used the first five stages of the dataset mentioned in Chapter 5. The dataset was split into training, validation, and test parts, consisting of 70%, 15%, and 15% of the dataset.

7.1.2 Image augmentations

The image was augmented by a single pipeline that applied the following transformations with corresponding probabilities p.

- Normalize the image by subtracting the dataset's mean and dividing by the standard deviation of the dataset (mean$= 0.37$, std$= 0.28$), $p = 1$.
- Resize and pad to $1024 \times 1024/896$, $p = 1$.
- Horizontal flip, $p = 0.5$.
- Vertical flip, $p = 0.5$.
- Rotation, $p = 0.3$, rotation limit$= 10°$.
- Translation, $p = 0.5$, translation limit$= 10\%$ of the image size.
- Gaussian blur flip, $p = 0.3$, kernel size from 7 to 31.
- Gamma correction, $p = 0.3$, γ in range from 0.6 to 1.4.

The results of those augmentations can be observed in Appendix C.3.

We selected the dimensions to which the image will be resized based on some architectures' limitations, which require the input to be divisible by 128. In the beginning, we used square images in order to ease the implementation. Following that, we switched to rectangular images; however, no improvement in measured metrics was observed. Based on the architecture, we noticed a decrease in the GPU memory usage from 5% to 10%.

Computing power

All the computations were realized on the CMP cluster, consisting of multiple GPU nodes, and the experiments were conducted on Boruvka and Zorn machines. They both have 32 CPU cores, 256GB of RAM memory, and 8 NVIDIA GeForce GTX 1080-Ti graphics cards with 12GB of dedicated memory.

7.1.3 Neural network models

Multiple architectures of neural networks were used. As the work on the thesis progressed, the amount grew larger. Thus, a wider variety of models can be seen in the advanced stages of the dataset. We stopped using the YOLOv3 model after Stage two since the YOLOv5 model performed superiorly in stages one and two, see Tables 8.1 and 8.2.

All the models we used and their respective backbones are listed below.

- YOLOv3 with Darknet-53 backbone.
- YOLOv5 with the sixth generation of backbones. We used the small, medium, large and extra-large versions of those backbones. In further text, we will denote them as s6, m6, l6, and x6.
- Faster-RCNN with Resnet50 and Resnet101 backbone; R50 and R101 abbreviations will be used.

model-backbone	batch size	model-backbone	batch size
YOLOv5-s6	16	EfficientDet-D0	5
YOLOv5-m6	8	EfficientDet-D1	4
YOLOv5-l6	4	EfficientDet-D2	3
YOLOv5-x6	2	EfficientDet-D3	2
FRCNN-R50	2	EfficientDet-D4	1
FRCNN-R101	5	EfficientDet-D5	1
RetinaNet-swint	3	YOLOv3 - Darknet	4
RetinaNet-R50	4	-	-

Table 7.1: Maximal batch sizes that fit into 12GB GPU for a given model

- RetinaNet (RetN) with Resnet50 and a tiny swin transformer (swint) backbones.
- EfficientDet with D0,D1,D2,D3,D4 and D5 backbones.

Batch-size

The batch size differed based on the model's architecture and the selected backbone. We always used the biggest batch size able to fit into the graphic card's memory. An overview of batch sizes for different combinations of architectures and backbones is in Table 7.1. Note that those batch sizes allow for the accumulation of four forward passes into the GPU memory.

Optimizers

We used the Adam optimizer during all experiments. The parameters β_1, β_2 were set to 0.9 and 0.999 and weight decay was chosen to be 10^{-6}. Since we could not fit reasonably big batch sizes into the GPU, the optimization step was performed every four forward passes. This should emulate a bigger batch size and increase the chance of finding a global optimization minimum.

Learning rate (LR)

First, we experimented with different learning rates and used the LR Range Test to find the initial LR. There was no difference in the final performance of the model, and the number of epochs required to train the model did not differ significantly. The test's LR did not lead to model convergence in rare cases. We, therefore, selected a constant initial LR of 10^{-4} and used a ReducedLROnPlateau scheduler. The value monitored by the scheduler was the validation loss, and the LR decreased by a factor of 5 when the improvement of the loss stalled for five consecutive epochs.

Termination condition

The training was halted when the $AP@.5$ did not increase throughout ten epochs, but not earlier than after 50 epochs from the beginning of the training.

7.2 Improvements

In this section, we propose improvements to the training protocol as well as a change to models trained with small batch sizes.

7.2.1 Training protocol changes

We replaced the Adam optimizer by AdamW with the same β_1 and β_2. Furthermore, a CosineAnnealingLR rate scheduler was used. The half-period of cosine was set to 70 epochs and the minimal LR to 10^{-7}.

We trained multiple models from Section 7.1.3 on the stage five dataset with this setting.

7.2.2 Group normalization

As mentioned in Section 3.5.3, batch normalization is superior to group normalization when used with batch sizes greater than eight. Since the size of EfficientDet-D4 and D5 models allowed us to use batch-size of one, we replaced all batch-normalization layers with group normalization. The channels per group parameter, for a given layer of group normalization, was set to 16[1], when the number of channels in the layer was divisible by 16. Otherwise, we selected one of the following values 2,4,8—prioritizing the higher of those.

The training was conducted according to the protocol described previously.

7.3 Model inspection

We conducted several experiments to assess the model's behavior. All experiments were done on the dataset created in the fifth stage with models, whose results are in Tables 8.6, B.4 and B.5.

7.3.1 Size of backbone

We explored the influence of the backbone choice on the model's performance. For this purpose, we used the YOLOv5 model and trained it multiple times with different backbones, including small, medium, and large backbones. The training protocol that was used was approached as described in Section 7.1, including the improvements mentioned in Subsection 7.2.1.

[1]Wu and He [40] showed this to be the best performing value when evaluated on ImageNet dataset.

7.3.2 Weight decay

We experimented with different values of weight decay. Eventually, we used the Faster-RCNN architecture with ResNet50 backbone and YOLOv5 with medium size backbone, and tested the following values of weight decay: $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$ on both models.

7.4 Model ensembling

In this section, we present our approach to model ensembling. First, we describe the process of unifying the predictions into the same format. We then propose an approach to finding optimal hyperparameters of the ensembling process. After that, we suggest improving the ensembling methods by including area awareness. In the end, we assess the importance of the diversity of architectures during the ensembling.

7.4.1 Data-format

For every model included in the ensembling, we load its trained weights and generate predictions on the dataset's training, test, and validation parts. When predicting, each image is rescaled to the size required by the corresponding model and normalized. This leads to predictions being in the space of the transformed image. Therefore we use inverse transformation to the rescaling to obtain coordinates in the original image. This allows us to combine predictions regardless of the model's architecture. We found the confidence threshold that maximized the F1 score and discarded all predictions with a confidence lower than that value. The confidence value was stored alongside the predictions.

Predictions were saved into JSON files to be reused without the need to generate new predictions. The format of the data is in Figure 7.1.

Confidence values maximizing F1 score differ (see Table B.2) across the models m, we therefore normalize them by the following formula:

$$s_{j,i} = \frac{s_{j,i} \max_l S_l}{S_j}, \quad j \in \{1, ..., m\}, i \in 1, ..., n_j \tag{7.1}$$

Note that Formula 7.1 is depended on the models included in the ensemble process, and thus weights cannot be normalized before saving them to the JSON file.

Manual tuning

We performed the ensembling by the approach described in Section 3.10 with the models mentioned in Table 8.4. The ensembling method was WBF. We estimated the weights for ensembling from the results of the individual models, and for the threshold value T, we selected values proposed by the authors of the WBF ensembling method [49].

```
{    "confidence threshold" : T
    "filename" : {
    "bboxes" : [[x1, y1, x2, y2],...],
    "labels": [l1, l2,...],
    "scores": [s1, s2,...],
    "stage" : "test" / "val" / "train"
    },
 "filename2" : {...},
 ...
}
```

Figure 7.1: Structure of the data in .json file used to store model predictions

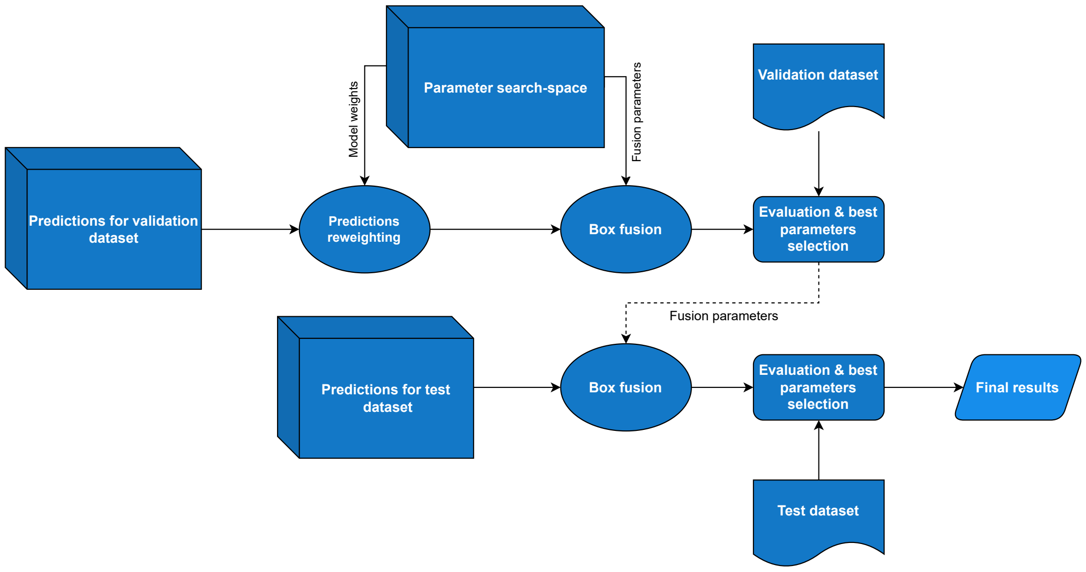


Figure 7.2: Schema of the search of hyperparameters and weights for ensembling

Grid-search

Grid search over hyperparameters was performed in the following manner: Parameter search space was defined as can be seen in Table 7.2. We evaluated $AP@.5$ on every image of the validation dataset and averaged those values. The best hyperparameters were selected and used for evaluation on the test dataset. The whole workflow is shown in Figure 7.2.

7.4.2 Area-aware ensembling

We proposed a change to the weighting function in ensembling; see e Equation 3.20. The new weighting function described in Equation 7.2 is aware of the area of the ensembled boxes. We hypothesized that this would increase the modeling capacity of the ensemble methods.

Parameter	minimal value	maximal value	step
Model weight	0.12	1	0.22
IOU T	0.3	0.9	0.05
Sigma	0.3	1	0.1

Table 7.2: Hyperparameter search-space for area-aware model ensembling

Parameter	minimal value	maximal value
Model weight (small, medium, large)	0.01	1
IOU T	0.05	1

Table 7.3: Hyperparameter search-space for model ensembling

The Equation 3.20 was modified to:

$$\mathcal{S} = \bigcup_{i=1}^{M} \frac{S_i f(B_i, \mathbf{w_i})}{F}, \quad f(B_i, \mathbf{w_i}) = \begin{cases} w_1, & \text{area}(B_i) \leq 32^2 \\ w_2, & 32^2 < \text{area}(B_i) \leq 96^2 \\ w_3, & 96^2 < \text{area}(B_i) \end{cases} \tag{7.2}$$

We tried to perform a grid search with the Equation 7.2 used for the weighting of boxes, but the number of parameters grew exponentially, making the grid search computationally untractable. Therefore, we used the Optuna optimization library to perform this task. The search space is in Table 7.3; please note that the search space became continuous contrary to the previous.

7.4.3 Assessing the importance of different models in ensembling

To get an insight into what affects the performance of model ensembling, we compare the following: an ensemble of multiple models with the same architecture and backbone, an ensemble of models with the same architecture and different backbones, and an ensemble of different architectures. All models selected for ensembling were trained on the stage five dataset, including the changes proposed in Section 7.2.1. The results of a subset of those models are in Table 8.6.

After selecting the models, we ensembled them using the WBF method. Model weights and IOU threshold T value were optimized on the validation dataset as described in Section 7.4.1. The Optuna optimization software was used to find those, where we set the search space to be in the range from 0 to 1 for all model weights as well as for the IOU threshold T. The optimization process was terminated after 3000 trials.

After obtaining the ensembles, we assess the importance of each model included in the ensemble by a method based on functional analysis of variance (FANOVA) [75]. We used an already implemented solution that is a part of the Optuna library [76].

The same architecture and backbone

We used the YOLOv5 architecture with a medium sized backbone, since multiple models were trained for experiments mentioned in Section 7.3. The summary of used models is available in Table 7.4.

The same architecture different backbones

We used the YOLOv5 architecture again but included two models with a small backbone, three with a medium-sized backbone, and three with the large backbone. When choosing those, we tried to match the performance of the selected models with those chosen previously. This was done to ensure the maximal comparability of those two ensembles.

Different architecture

Here, we used the best model we had at our disposal. The ensemble consisted of the following models: Two YOLOv5-large, two YOLOv5-medium, YOLOv5-small, EfficientDet-D1, EfficientDet-D3, RetinaNet-swint, and Faster-RCNN-Resnet50. Even though we picked the best available models, the difference in $AP@.5$ against YOLOv5-all was 2\$, as can be seen in Table 7.4, where we denoted a group of models with different architectures as Mixture.

Experiment	num.models	mean	std	min	max
YOLOv5-m	8	0.696	0.014	0.668	0.719
YOLOv5-all	8	0.693	0.017	0.659	0.719
Mixture	10	0.707	0.014	0.676	0.725

Table 7.4: Comparison of the models involved in ensembling by statics of their $AP@.5$ metrics

7.5 Dental restorations segmentation

In this section, we propose a non-deep learning method for the segmentation of dental caries. We tune the hyperparameters of this approach to achieve the best possible performance. Following that, we switched our approach and trained a deep learning model called U-Net.

7.5.1 Non-deep learning approach

We decided to test the approach proposed by Abdalla-Aslan, and Yeshua [18, 17] as described in Section 4.2. This means that we defined a pipeline of image processing operations, where we:

- Thresholded the image: We tried Otsu's thresholding method and Gaussian blur with kernel size K_b applied prior to that. Furthermore, Gaussian and mean

Hyperparameter	minimal value	maximal value	step
K_t	33	83	10
K_b	1	26	5
T	1	15	2
K_d	41	81	20
K_o	1	36	5

Table 7.5: Hyperparameter search space for restorations segmentation pipeline

adaptive thresholding methods were evaluated, where we tested different kernel sizes K_t and threshold values T.

- Removed predicted pixels at the border of the X-ray image: Bitewing X-ray images usually do not have a rectangular shape, as illustrated in 2.2. Bitewing radiographs are therefore padded with black color to obtain the rectangular shape. This leads to a high contrast at the border of the radiograph, which gets detected by adaptive thresholding methods. Therefore, we detect this padding and morphologically dilate it by the square kernel of size $K_d \times K_d$. Border pixels obtained by the dilation are removed from the thresholded image.
- Applied morphological opening: We apply morphological opening to filter out falsely detected regions; we use a square-shaped kernel with size K_o.

The pipeline defined above has four hyperparameters that can be tuned and three thresholding methods. Consequently, we define a grid-shaped search space as can be observed in Table 7.5. Note that when we searched for the value of K_t, we did not search for K_b and vice versa. The dataset was split into two equally-sized parts, called tune and test. We evaluated the IOU metric on each image of the tune part of the dataset and averaged it for each set of hyperparameters. We selected the best hyperparameters based on the average IOU value and used those to evaluate on the test part of the dataset.

7.5.2 Deep-learning approach

Model training

The dataset was split into training, validation, and test parts with a 70:15:15 ratio. For the dataset's training part, we used augmentations described in Section 7.1.2. Resizing of images was removed from the augmentation pipeline, and for that reason we worked with full-sized radiographs. Images in the validation and test part of the dataset were only normalized. We used the U-Net architecture model as proposed by author [13], only changing the depth to five downscaling layers. As a loss function, we used Soft-dice loss. The LR value of 10^{-2} was used together with ReduceLROnPlateau scheduler. The model was trained for 50 epochs by the Adam optimizer. At the end of the training, we selected the best model according to IOU on the validation dataset and evaluated its performance on the test dataset.

Post-processing

We used post-processing methods inspired by the non-deep learning approach on the best-performing model. The image was first morphologically opened by a square kernel with a size of K_O and then closed by a kernel size of K_C.

Since there were two hyperparameters involved, we performed a grid-search, searching for the kernel of size from 1 to 41 with the step size of 4. Firstly, we saved predictions for all images in the validation dataset. This ensured that we did not have to pass the image through the U-Net model for each parameter. Post-processing with all defined kernel values was applied. The best-performing hyperparameters were selected, and post-processed predictions were evaluated on the test dataset.

7.5.3 Model training improvements

We tried the following changes to the model training pipeline:

- AdamW optimizer was applied instead of Adam.
- CosineAnnealingLR scheduler was used. The half-period of cosine was set to 40 epochs and the minimal LR was set to 10^{-7}.
- Binary cross-entropy loss was tested instead of Soft dice loss as well as a combination of Soft dice loss with BCE.
- We removed the maximal amount of epochs and instead used the IOU value on the validation set as a stopping criterion. Whenever we observed no improvement of IOU for ten epochs, the training was stopped.

Following the training, we performed the same post-processing. We found the best hyperparameters by the Optuna optimization library. The search space ranged from 1 to 41 for both kernels.

Chapter 8

Results

This chapter states the results of the proposed methods.

In Section 8.1, we show the results of the trained models on different stages of the dataset.

Section 8.2 contains results of improvements of the training pipeline proposed in Section 7.2.1.

In Section 8.3, we depict how different backbones' sizes and weight decay affected the model's performance.

Section 8.4 reports the influence of ensembling of neural networks. Note that in this section, we obtain the best-performing model.

Section 8.5 contains results of algorithms for segmentation of dental restorations.

In Section 8.6, we compare the results of our best-performing model with the literature.

In the final Section 8.7 of this chapter, we show figures to showcase the performance of our models.

8.1 Model comparison on different datasets

This section compares the performance of different model architectures and their backbones. It is divided into five subsections corresponding to five stages of the dataset. For more details about the dataset, see Section 5.1. For each of the five stages of the dataset, we report the average precision metric on the test part of the dataset. The training of all models was conducted according to the training protocol described in Section 7.1. Tables in this section use abbreviation described in Section 7.1.3.

8.1.1 Stage one dataset

In Table 8.1 the reader can see results obtained on the first stage of the dataset. None of the trained models performed well, especially the Faster R-CNN model achieved low average precision values. We attribute that to the inhomogeneity of the dataset (as described in Section 5.1) and to the low amount of data.

Model	AP	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
FRCNN-R50	0.045	0.168	0.0064	0.109	0.187	0.141
YOLOv3	0.078	0.238	-	-	-	-
YOLOv5-l6	0.082	0.258	0.02	0.211	0.309	0.327
YOLOv5-x6	0.087	0.268	0.04	0.204	0.324	0.302
EfficientDet-D4	0.081	0.242	0.007	0.198	0.234	0.287

Table 8.1: Comparison of the trained models on the stage one dataset

8.1.2 Stage two dataset

Even though the dataset grew in size by more than 50% since stage one, from Tables 8.1 and 8.2, we see that the YOLOv3 model improved by less than 10% . On the contrary, the performance of YOLOv5 model improved by circa 25%. Due to the low performance of YOLOv3 and its similarity with superior YOLOv5 architecture, we will not experiment with YOLOv3 in the following sections. In the last two rows of Table 8.2, we can see the discrepancy in the average precision when evaluated on the test and train part of the dataset. This ensures us that the model can fit the data well, and we only need to alleviate the generalization gap.

Model	AP	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
YOLOv3	0.093	0.258	-	-	-	-
YOLOv5-l6	0.097	0.318	0.03	0.281	0.310	0.378
YOLOv5-x6	0.105	0.337	0.05	0.278	0.323	0.392
EffDet-D4	0.089	0.296	0.01	0.272	0.291	0.342
EffDet-D4, train	0.421	0.839	0.362	0.758	0.852	0.801

Table 8.2: Comparison of the trained model on the stage two dataset

8.1.3 Stage three dataset

As mentioned in Section 5.1, there were no additional data added in this stage, but MDDr. Tichý did a review of the dataset as well as corrected any erroneous annotations. The dataset review increased the $AP@.5$ of the EfficiendDet-D4 model by 76%, as can be seen in Tables 8.3 and 8.2. This significant performance gain suggests that the low performance of models in Tables 8.1 and 8.2 was caused by errors in the dataset.

Performance of EfficentDet and YOLOv5 models evaluated on training part of dataset can be found in Table B.1.

Model	AP	$AP@.3$	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
YOLOv5-l	0.249	0.734	0.631	0.132	0.598	0.671	0.607
EffDet-D4	0.168	0.666	0.525	0.041	0.435	0.606	0.527

Table 8.3: Comparison of trained models on the test part of stage three dataset

8.1.4 Stage four dataset

The average precision of models trained on this dataset stage is in Table 8.4. Furthermore, precision-recall values for confidence threshold maximizing F1 score are in Table B.2, which is located in Appendix. Even though we introduced multiple new architectures and revisited Faster R-CNN with two different backbones, the performance gain obtained in this stage was not as prominent as the one observed when moving from stage three to stage four.

In Table 8.4, we can observe how different architectures differ in their performance on small, medium-sized, and large boxes. Comparing YOLOv5-m6 and RetinaNet-ResNet50 models shows that their overall performance (measured by $AP@.5$) almost matches, but when comparing their $AP@.5_L$ metrics, we see a 15% difference. We try to exploit this behavior by the approach described in Section 7.4.2.

Model	AP	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
FRCNN-R101	0.285	0.675	0.198	0.568	0.717	0.772
FRCNN-R50	0.284	0.658	0.204	0.557	0.695	0.77
YOLOv5-m6	0.288	0.644	0.209	0.593	0.667	0.766
YOLOv5-l6	0.284	0.644	0.203	0.551	0.701	0.612
EffDet-D4	0.251	0.605	0.15	0.49	0.677	0.545
RetN-swint	0.266	0.66	0.175	0.497	0.721	0.786
RetN-R50	0.263	0.643	0.174	0.547	0.696	0.663

Table 8.4: Performance comparison of multiple models trained on the stage four dataset

8.1.5 Stage five

The results in Table 8.5 show a steady improvement compared to those in Table 8.4. We see that YOLOv5-l6 achieved worse results than in stage four. This result is not emphasized since we believe that if trained multiple times, the results of this architecture would eventually improve. On the contrary, we observe that the EfficientDet-D4 model lagged behind YOLOv5 in all stages of the dataset. Therefore, in Section 7.2.2 we experiment with usage of group normalization.

Model	AP	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
FRCNN-R101	0.328	0.71	0.263	0.613	0.742	0.816
FRCNN-R50	0.334	0.715	0.273	0.595	0.757	0.809
YOLOv5-m6	0.346	0.708	0.284	0.622	0.744	0.754
YOLOv5-l6	0.295	0.625	0.232	0.533	0.691	0.489
EffDet-D4	0.288	0.648	0.219	0.548	0.699	0.655
RetN-swint	0.328	0.72	0.241	0.565	0.776	0.775

Table 8.5: Performance comparison of multiple models based on mean average precision metrics

8.2 Improvements

8.2.1 Training protocol improvements

The average precision of models trained with incorporated improvements proposed in Section 7.2.1, can be seen in Table 8.6. Furthermore, average recall values can be observed in Table B.4, which is located in the Appendix, together with a table of precision-recall values for a given confidence thresholdB.5.

Even though the best performing model in Table 8.6 improved negligibly over the best performing one in Table 8.5. We notice that models achieved better results on average along with more stable training. In this stage, we newly used YOLOv5-s and EfficienDet-D1. Despite both being low-parameter networks, they performed almost on par with others.

Model	AP	$AP@.3$	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
YOLOv5-l6	0.347	0.796	0.725	0.291	0.597	0.772	0.753
YOLOv5-m6	0.343	0.795	0.719	0.287	0.636	0.752	0.785
YOLOv5-s6	0.327	0.79	0.697	0.281	0.559	0.739	0.826
Effdet-D1	0.319	0.787	0.701	0.251	0.584	0.752	0.808
FRCNN-R50	0.311	0.788	0.705	0.231	0.629	0.737	0.788
FRCNN-R101	0.316	0.792	0.688	0.239	0.563	0.732	0.793
RetN-swint	0.325	0.803	0.723	0.249	0.579	0.78	0.758

Table 8.6: Comparison of AP values between different models trained by an improved training protocol

8.2.2 Group normalization

Chart of $AP@.5$ for EfficientDet-D4 models with batch-normalization layers and group normalization layers can be found in Figure 8.1. Model using batch normalization achieved $AP@.5$ of 0.634 on the test dataset, outperforming the model using group normalization with $AP@.5 = 0.694$. Despite the performance increase induced by group normalization, the EfficientDet-D4 model performed comparably with the models in Table 8.6. Therefore, we stopped using this model further on, because the training of this model was also more computationally demanding than the rest of the models.

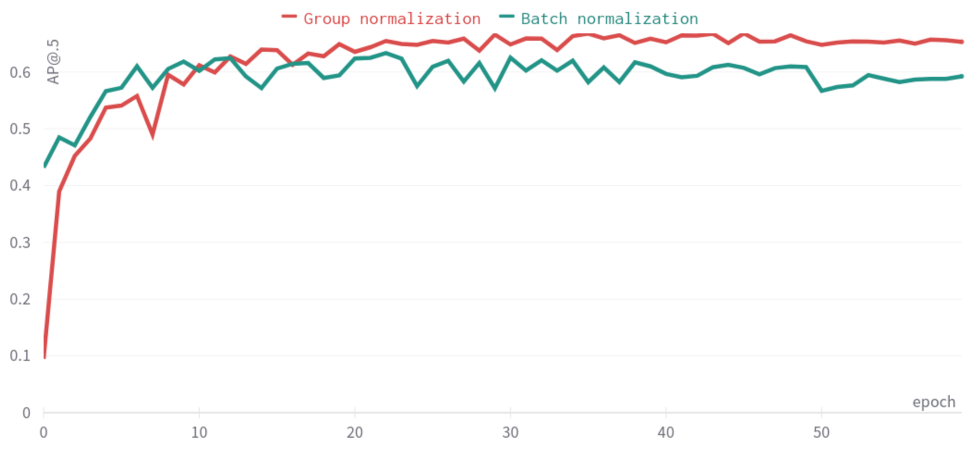


Figure 8.1: Difference in $AP@.5$ amog EfficientDet-D4 model with batch normalization and group normalization

8.3 Model inspection

8.3.1 Size of backbone

The table 8.7was computed from the statistics of 29 models, where each type of backbone was represented by 8 to 12 models. The columns mean, std, max and min denote statistics of $AP@.5$ obtained by trained models on the test set. Furthermore, we can inspect the size of a given backbone, which is induced by its number of parameters (Par) and floating-point operations FLOPs. The last column of Table 8.7 shows the average time required to train the given backbone for 60 epochs.

Backbone	Mean	Std.	Max	Min	Par[M]	FLOPs[G]	Time[h]
Small	0.68	0.0197	0.651	0.697	12	21	2.1
Medium	0.696	0.0126	0.669	0.719	35	63	3.5
Large	0.703	0.0136	0.681	0.725	76	141	5.2

Table 8.7: Comparison of $AP.@5$ metric for different backbones of YOLOv5 architecture

8.3.2 Weight decay

In Figures 8.2 and 8.3, we see the comparison of Faster-RCNN and YOLOv5 models using different weight decay. The results obtained by evaluating trained models on the test part of the dataset did not differ across the values of weight decay.

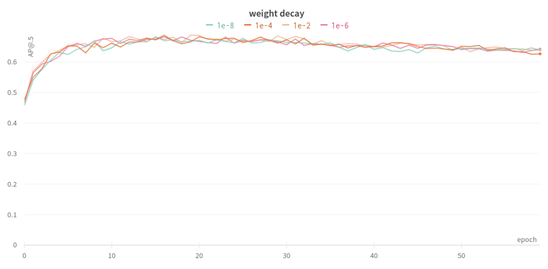


Figure 8.2: $AP@.5$ of Faster-RCNN model with varying weight decay values. The metric is computed on the validation part of the dataset during training.

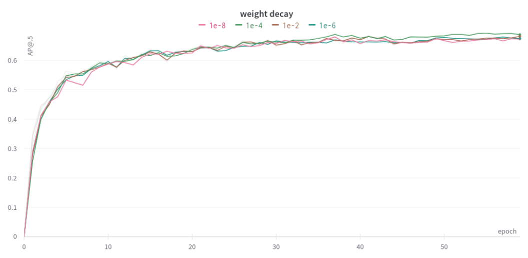


Figure 8.3: $AP@.5$ of YOLOv5 model with varying weight decay values. The metric is computed on the validation part of the dataset during training.

8.4 Ensembling

In this section, we show the performance of model ensemblings with handpicked parameters 8.4.1as well as ensemblings obtained by parameters found by a grid-search. Furthermore, we report how the diversity of models involved in the ensembling affects its results.

8.4.1 Manually-picked parameters

In Table 8.8, the reader can see the results obtained by handpicking circa ten sets of hyperparameters based on our qualified guess. We then evaluated the hyperparameters on the validation part of the dataset, and the best-performing ones were selected and evaluated on the test part of the dataset. The first group (G1) contained the following models trained on the stage four dataset: RetinaNet-swint, YOLOv5-m, and RetinaNet-ResNet50. The second group (G2) was composed of Faster R-CNN-Resnet101, YOLOv5-m, and RetinaNet-swint. All of those models were trained on stage four of the dataset. From the Tables 8.8 and 8.4, we can infer that ensembling of models improved $AP@.5$ by 3% over the best performing model included in the ensembling.

Models	AP	$AP@.3$	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
G1	0.303	0.776	0.694	0.216	0.605	0.729	0.803
G2	0.305	0.783	0.695	0.218	0.598	0.733	0.807

Table 8.8: WBF ensembling of multiple models, where we handpicked the parameters of the ensembling process. The models were trained on the stage four dataset.

8.4.2 Grid search results

All models included in the ensembling were trained on the stage-five dataset. Therefore, their results are in Table 8.5.

The best hyperparameters for a given ensembling method found by a grid search are in Table 8.9. We omitted parameter σ used only in S-NMS from the table. Its optimal value, according to the grid search, was 0.8. The average precision of models ensembled with parameters from Table 8.9 is available in Table 8.10 and average recall values are in Table B.6. Precision and recall values based on the confidence threshold that maximizes F-score can be seen in Table B.7. In Tables 8.9 and 8.10, we used notations introduced in Section 3.10. By WBF-A, we refer to the method proposed in Section 7.4.2, where S,M,L mean weights for small, medium-sized, and large boxes.

Method	FRCNN R50	YOLOv5 m6	RetN swint	FRCNN R101	T
NMS	1	0.4	0.4	0.85	0.6
SNMS	1	0.12	0.12	0.12	0.7
NMW	0.85	0.25	0.70	0.85	0.45
WBF	1	0.4	0.85	0.85	0.65
WBF-A S	0.94	0.31	0.98	0.72	0.64
WBF-A M	0.77	0.47	0.85	0.69	0.64
WBF-A L	0.84	0.31	0.88	0.91	0.64

Table 8.9: Hyperparameter values of ensembling methods found by a grid-search

Method	AP	$AP@.3$	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
NMS	0.346	0.818	0.735	0.28	0.618	0.775	0.829
SNMS	0.348	0.807	0.722	0.295	0.609	0.758	0.819
NWM	0.364	0.829	0.759	0.302	0.641	0.802	0.854
WBF	0.378	0.832	0.77	0.323	0.663	0.807	0.875
WBF-A	0.376	0.832	0.768	0.318	0.651	0.806	0.875

Table 8.10: Average precision of models ensembled with parameters from Table 8.9

8.4.3 Assessing the importance of different models

In this section, we present ensembling results based on the approach from Section 7.4.3. In Tables 8.11, 8.12 and 8.13, we see results of ensembles composed of different models. Note that the ensembling of varying architectures (Mixture) achieved the best results out of all models evaluated in this thesis. We will use this model to compare our results with related publications.

From Table 8.11, we can see that the usage of varying backbones increases the gain in average precision by circa 1.5%, when compared to the ensembling with the identical backbones. Furthermore, using different architectures increased the performance by an additional 3.2%. Please note that models included in ensembling with different architectures had an average $AP@.5$ by 2% higher than models included in the former two ensembling approaches, see Table 7.4. If we adjust the results for that, we expect a 1.2% gain solely from using models with varying architecture.

The importance of individual models in the ensembling is in Figure 8.4 for Mixture architectures and in Figures C.5, C.6 located in Appendix for the remaining two groups of models.

Models	AP	$AP@.3$	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
Mixture	0.389	0.838	0.774	0.338	0.665	0.811	0.876
Y5-mix	0.379	0.819	0.75	0.34	0.636	0.79	0.84
Y5-m	0.368	0.812	0.741	0.329	0.648	0.775	0.844

Table 8.11: Average precision of ensemble models

Models	AR	$AR@.5_{10}$	$AR@.5$	$AR@.75$	$AR@.5_S$	$AR@.5_M$	$AR@.5_L$
Mixture	0.586	0.917	0.978	0.582	0.946	0.991	0.991
Y5-mix	0.579	0.911	0.959	0.599	0.929	0.972	0.972
Y5-m	0.562	0.906	0.949	0.572	0.909	0.964	0.964

Table 8.12: Average recall of models ensembled by parameters from Table 8.9

Models	Precision	Recall	F-score	Confidence threshold
Mixture	0.751	0.7	0.725	0.294
YOLOv5-mix	0.728	0.69	0.708	0.241
YOLOv5-m	0.726	0.67	0.697	0.272

Table 8.13: Precision, recall, and F-score based on the confidence threshold for different ensembling methods

8.5 Dental restorations segmentation

In the following section, we report results obtained by a non-deep learning pipeline for dental restorations segmentation proposed in Section 7.5.1, as well as deep learning model U-Net trained according to the description in Section 7.5.2.

8.5.1 Non-deep learning approach

Hyperparameters ensuring the best performance on the validation part of the dataset are in Table 8.14. The performance of the pipeline given the parameters in Table 8.14 on the test dataset can be seen in Table 8.15. The segmentation of the image together with output after each auxiliary stage is in Figure 8.5.

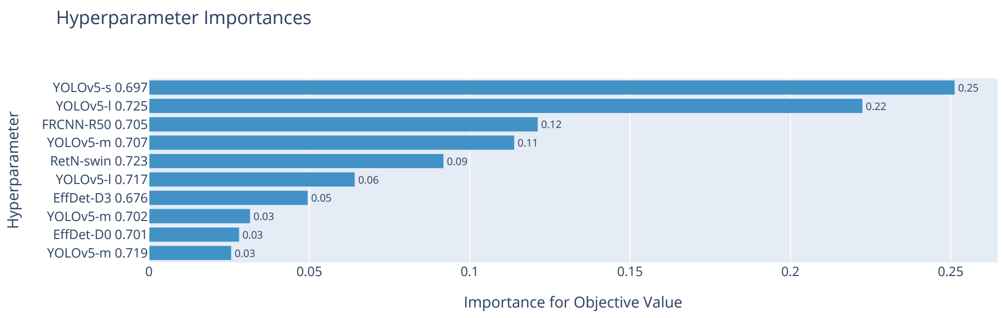
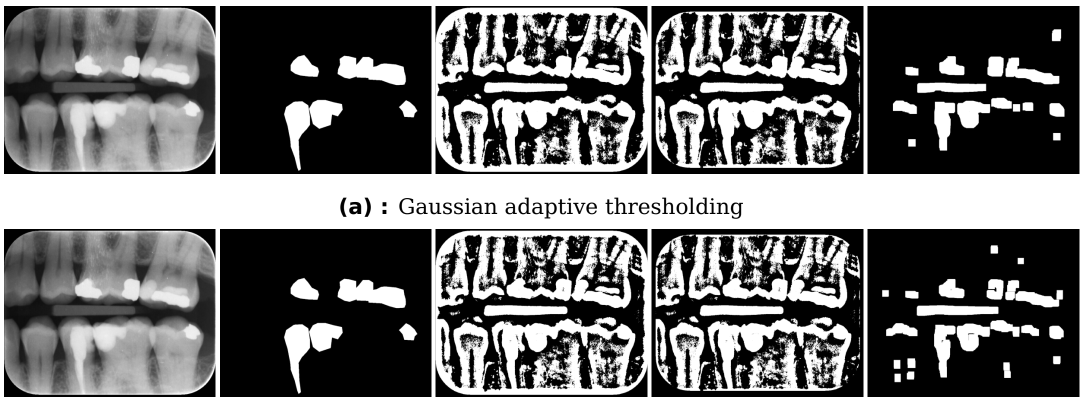


Figure 8.4: Importance of different models during ensembling with different architectures



(a) : Gaussian adaptive thresholding

(b) : Mean adaptive thresholding

Figure 8.5: From the left: X-ray image, ground-truth pixel mask, thresholded image, removal of border pixels, morphological opening

Hyper-parameter	Adaptive mean	Adaptive Gaussian	Otsu's
K_t	71	83	-
T	3	3	-
K_d	41	41	61
K_o	31	36	36
K_b	-	-	21

Table 8.14: Best hyper-parameters for non-deep learning pipeline

Model	Dice	IOU
Adaptive mean	0.364	0.314
Adaptive Gaussian	0.328	0.274
Otus's thresholding	0.102	0.088

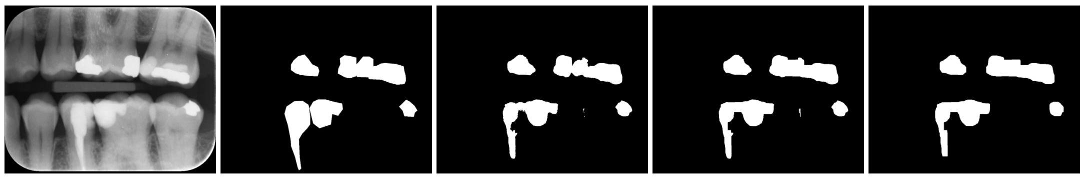
Table 8.15: Results of non-deep learning approach to dental restorations segmentation given the hyperparameters in Table 8.14

8.5.2 U-Net

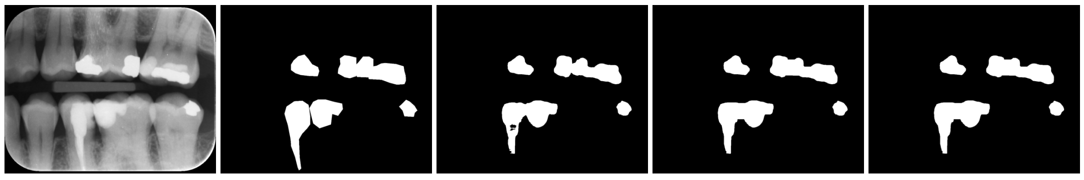
Two U-Net models were trained with the settings proposed in Section 7.5.2. We will call them U-Net-baseline (U-Net-B) and U-Net-improved (U-Net-I). With the letters PP, we denote that the model's output was post-processed by the approach described in Section 7.5.2. In Figure 8.7, we see IOU evaluated on the validation dataset throughout the model training, where each line corresponds to a different loss function. We notice that a combination of BCE and soft-Dice loss achieved the highest IOU results, while soft-Dice loss diverged after a promising performance at the beginning of the training.

The best hyperparameters for post-processing of both models are in Table 8.16. In Figure 8.8, we observe how the choice of different hyperparameters for the post-processing pipeline affected the performance of U-Net-B model.

Figure 8.6 shows the segmentation results and compares those with the ground truth mask.



(a) : Baseline U-Net model



(b) : Improved U-Net model

Figure 8.6: From the left: X-ray image, ground-truth pixel mask, the output of the model, output processed by morphological opening, output post-processed by morphological opening and closing.

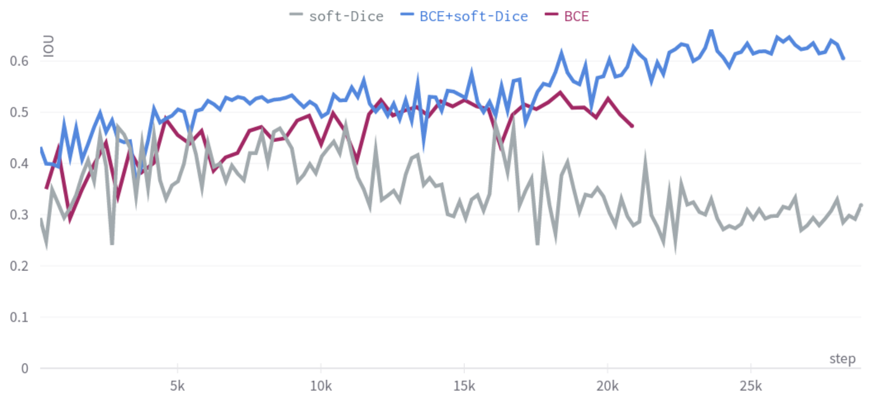


Figure 8.7: IOU throughout the training of U-Net model for different loss functions

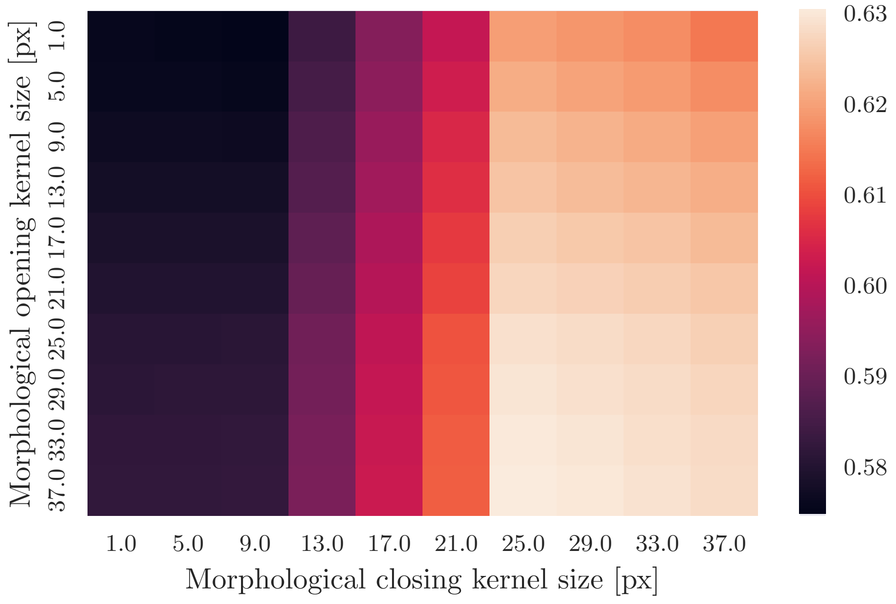


Figure 8.8: Value of IOU metric based on the size of kernels K_o, K_c in morphological operations that were used for post-processing

Parameter	K_o	K_c
U-Net	37	25
U-Net-I	33	5

Table 8.16: Optimal parameters for model post-processing found by a grid-search

Model	Dice	IOU
U-Net-B	0.663	0.575
U-Net-B-PP	0.714	0.623
U-Net-I	0.747	0.662
U-Net-I-PP	0.760	0.676

Table 8.17: Results of U-Net models

Author	Precision	Recall	F1-Score	Accuracy	$AP@.5$
This thesis	0.751	0.7	0.725	0.726	0.774
Srivastava et al. [16]	0.615	0.805	0.7	-	-
Kumar & Srivastava [19]	0.7	0.53	0.614	-	-
Bayrakdar et al. [56]	0.78	0.77	0.78	-	-
Bayraktar et al. [62]	-	0.72	-	0.946	0.872
Cantu et al. [55]	-	0.75	0.73	0.8	-

Table 8.18: Comparison of results of this thesis with results in related publications

8.6 Comparison of results with related publications

In Table 8.18, we see a comparison of caries detection results of this thesis contrasted to results achieved by related works. We compared with only those who selected a similar approach to ensure at least a minimal amount of comparability.

Note that Cantu et al. [55] solved the problem of dental caries localization as a semantic segmentation task. The recall and F1-score are calculated per pixel, while others worked with bounding boxes. However, the reader can still estimate how their work compares to others in table.

8.7 Visualization of models

All figures in this section were generated by the best performing ensemble model for caries detection, introduced in Section 8.11, and the U-Net-I model (without post-processing).

Figure 8.9 shows the relationship between the number of false positives per image and the recall of the model. The graph was truncated and did not include points for $recall > 0.91$ since it would decrease the chart's readability.

Figure 8.11 overlays the bitewing image with predictions of both caries detection and restorations segmentation models. For more similar figures, see Appendix B.1 and C

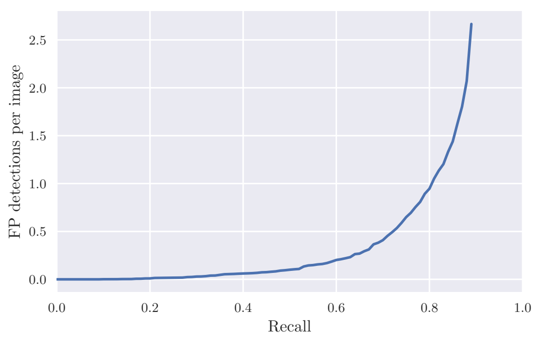


Figure 8.9: Number of false positives per image for a given value of recall

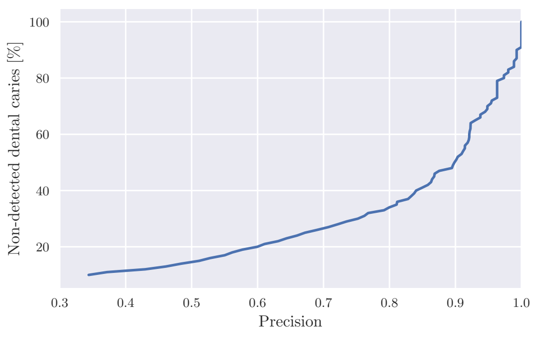
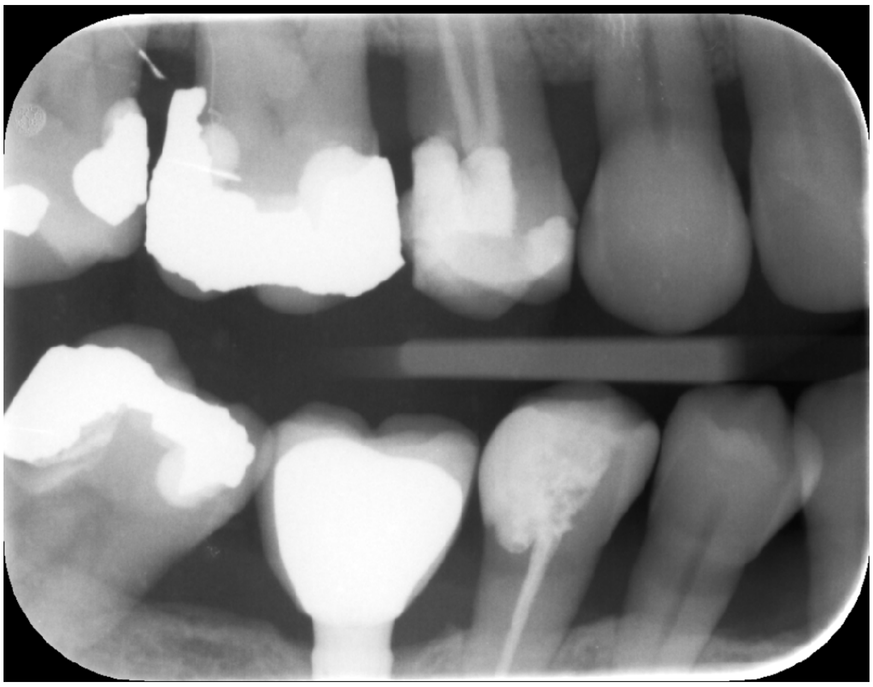
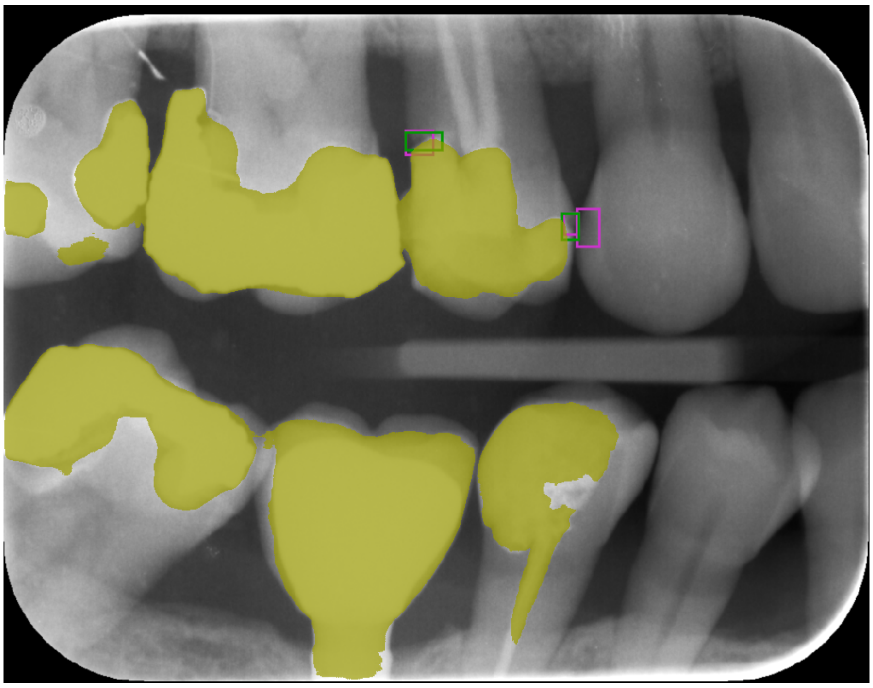


Figure 8.10: Percentage of nondetected dental caries based on the precision of the model



(a) : Original image



The o

(b) : Output of our models

.

Figure 8.11: Segmented dental restorations in yellow, predicted dental caries in pink and ground truth of dental caries in green. We see a single false positive detection on the top right of the image. The author of the dataset acknowledges it to be a missing ground truth label

Chapter 9

Discussion and further suggestions

9.1 Comments on model comparison and their improvements

We observed that updating the dataset by either increasing the number of data or correcting annotations mistakes showed a significant improvement in the monitored metrics.

The results differ only in units of percent if we use models with smaller backbones, as shown in Table 8.7. Furthermore, parameter-heavy models such as EfficientDet-D4 performed worse than others, see Table 8.4. This was surprising since EfficentDet-D4 was out-performing all other tested models on the MS COCO benchmark [24, 8].

The weight decay did not affect the behavior of YOLOv5 and Faster R-CNN. This was unexpected since we would anticipate an increase in performance on the validation and test dataset.

The use of group normalization significantly decreased the performance gap between EfficiendDet and other models (Section 8.2.2), but the results were still inferior to most of the models.

9.2 Ensembling

Model ensembling improved the results more than we anticipated. We assess that this was caused by the array of object detection models used for the ensemble. The results in Section strengthen this introduction8.4.3. We were surprised that our proposed aware ensembling solution (see Section 7.4.2) did not improve the results.

9.3 Dental restorations segmentation

9.3.1 Non-deep learning approach

The grid search did not find a hyper-parameter, ensuring that adaptive thresholding methods would detect only dental restoration. From Figure 8.5 we see that adaptive thresholding methods include many false-positive predictions. A significant amount of those is removed by morphological operations, but the IOU 0.314 is still relatively low compared to U-Net models. This corresponds to the results of Abdalla-Aslan et al. [18], who achieved a precision of 0.33 when segmenting dental restorations from panoramic images.

9.3.2 U-Net

The baseline U-Net model already showed significant performance gain over the non-deep learning segmentation pipeline. It was improved by more than 10% using morphological operations post-processing.

Improvements in the training process increased the performance of U-Net significantly; on the contrary, post-processing the improved U-Net model improved all tracked metrics negligibly.

In the figure 8.6, we compare the predicted pixel mask with the ground truth. In the border areas of dental restorations, the model seems to estimate its position better than the ground truth labels. This happens due to the unease of labeling data for segmentation tasks. The annotator needs to include many points in the bounding rectangle to correspond to the absolute position of the restoration. We believe that this hurts the reported performance of the model.

9.4 Comparison of results with related publications

Results obtained by this work have beaten those achieved by Srivastava at al. [16] and Kumar and Srivastava [**?**]. Please note that even though Kumar continued on the work published by Srivastava et al., even extending the dataset used by Srivastava twofold, all metrics reported by them dropped significantly. We cannot explain what caused this decrease in performance. Kumar does not address this problem in the published paper.

Bayrakdar2021 et al. [56] and Bayraktar2021 et al [62] reported better results than we achieved throughout our work. We find that surprising since both works had a significantly smaller amount of data (621 and 1000 images). We experimented with the YOLOv3 architecture used by Bayraktar202 [62], but in our experiments, it achieved worse results than other architectures. We, therefore, found their results to be irreproducible.

Cantu et al. achieved a similar F1-Score as we did in our work. The comparability of those results is limited since they solved semantic segmentation tasks contrary to object detection. F

Chapter 10
Conclusion

This thesis has developed a solution based on convolutional neural networks for dental caries detection and dental restorations segmentation. The best-performing model for dental caries detection achieved $AP@.5 = 0.725$, and an ensemble of ten models improved the results to $AP@.5 = 0.774$. The best model for dental restorations segmentation achieved an IOU of 0.676 and a Dice score of 0.76.

We contributed to creating a dataset containing 3989 bitewing images with 7257 annotated dental caries. Furthermore, for 521 images, a pixel mask with highlighted dental restorations is available. To our knowledge, this is one of the most extensive datasets created for caries detection.

During the dataset's creation, the model already proved to detect dental caries overlooked by a dentist, and since then, the model has improved significantly. Therefore, we believe that the model in its current state would be helpful during the diagnosis of dental caries. It could serve as a second opinion for the dentists that they could compare his beliefs against.

Further work

The primary focus should be on finishing the sixth stage of the dataset in the future. We believe that there are still overlooked dental caries, despite the decreasing number of caries in stage three. We believe that including additional

models with different architectures could further increase the performance. For example, parameter-heavy models such as EfficientDet with D4+ backbones could be trained and added to the ensemble. However, this would require a GPU with more dedicated memory. According to our experience, a 40GB GPU would be required to fit EfficientDet-D4 with batch size four into the GPU.

It may be worth exploring the option of backbone sharing across multiple tasks. The model for segmentation of dental restorations could benefit from the backbone shared with the model for object detection, which we trained on a significantly larger amount of data. Therefore, we believe that the backbone would be able to extract better features from the image and thus improve segmentation performance.

tj

Appendix A

Bibliography

[1] A. Creanga, H. Geha, V. Sankar, F. Teixeira, C. McMahan, and M. Noujeim, "Accuracy of digital periapical radiography and cone-beam computed tomography in detecting external root resorption," *Imaging science in dentistry*, vol. 45, pp. 153–8, Sep. 2015.

[2] "What is a Panoramic X-Ray," Sep. 2017. [Online]. Available: https://www.minthilldentistry.com/panoramic-x-ray

[3] J. Cowton, I. Kyriazakis, and J. Bacardit, "Automated individual pig localisation, tracking and behaviour metric extraction using deep learning," *IEEE Access*, vol. 7, pp. 108 049–108 060, 2019.

[4] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, jul 2020.

[5] G. Zhang and H. Li, *Effectiveness of Scaled Exponentially-Regularized Linear Units (SERLUs)*, Jul. 2018.

[6] K. Yin, *Sign Language Translation with Transformers*, Apr. 2020.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020.

[8] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *CoRR*, vol. abs/1911.09070, 2019. [Online]. Available: http://arxiv.org/abs/1911.09070

[9] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Mmdetection: Open mmlab detection toolbox and benchmark," 2019.

[10] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," 2017.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 10 012–10 022.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” 2017.

[13] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

[14] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, “Soft-nms – improving object detection with one line of code,” 2017.

[15] S. Lee, S. il Oh, J. Jo, S. Kang, Y. Shin, and J. won Park, “Deep learning for early dental caries detection in bitewing radiographs,” *Scientific Reports*, vol. 11, no. 1, aug 2021.

[16] M. M. Srivastava, P. Kumar, L. Pradhan, and S. Varadarajan, “Detection of tooth caries in bitewing radiographs using deep learning,” *CoRR*, vol. abs/1711.07312, 2017. [Online]. Available: http://arxiv.org/abs/1711.07312

[17] T. Yeshua, Y. Mandelbaum, R. Abdalla-Aslan, C. Nadler, L. Cohen, L. Zemour, D. Kabla, O. Gleisner, and I. Leichter, “Automatic detection and classification of dental restorations in panoramic radiographs,” *Issues in Informing Science and Information Technology*, vol. 16, pp. 221–234, 2019.

[18] R. Abdalla-Aslan, T. Yeshua, D. Kabla, I. Leichter, and C. Nadler, “An artificial intelligence system using machine-learning for automatic detection and classification of dental restorations in panoramic radiography,” *Oral Surgery, Oral Medicine, Oral Pathology and Oral Radiology*, vol. 130, no. 5, pp. 593–602, nov 2020.

[19] P. Kumar and M. M. Srivastava, “Example mining for incremental learning in medical imaging,” 2018.

[20] N. J. Kassebaum, E. Bernabé, M. Dahiya, B. Bhandari, C. J. L. Murray, and W. Marcenes, “Global burden of untreated caries: A systematic review and metaregression,” *J Dent Res*, vol. 94, no. 5, pp. 650–658, Mar. 2015. [Online]. Available: https://doi.org/10.1177/0022034515573272

[21] D. e. a. James, Spencer L. Abate, “Global, regional, and national incidence, prevalence, and years lived with disability for 354 diseases and injuries for 195 countries and territories, 1990–2017: a systematic analysis for the global burden of disease study 2017,” *The Lancet*, vol. 392, no. 10159, pp. 1789–1858, nov 2018.

[22] M. Hung, M. S. Lipsky, R. Moffat, E. Lauren, E. S. Hon, J. Park, G. Gill, J. Xu, L. Peralta, J. Cheever, D. Prince, T. Barton, N. Bayliss, W. Boyack, and F. W.

Licari, “Health and dental care expenditures in the united states from 1996 to 2016,” *PLOS ONE*, vol. 15, no. 6, p. e0234459, jun 2020.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[24] “Papers with Code - Browse the State-of-the-Art in Machine Learning.” [Online]. Available: https://paperswithcode.com/sota

[25] A. Rodriguez-Ruiz, K. Lång, A. Gubern-Merida, M. Broeders, G. Gennaro, P. Clauser, T. H. Helbich, M. Chevalier, T. Tan, T. Mertelmeier, M. G. Wallis, I. Andersson, S. Zackrisson, R. M. Mann, and I. Sechopoulos, “Stand-alone artificial intelligence for breast cancer detection in mammography: Comparison with 101 radiologists,” *JNCI: Journal of the National Cancer Institute*, vol. 111, no. 9, pp. 916–922, mar 2019.

[26] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, “Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network,” *Nature Medicine*, vol. 25, no. 1, pp. 65–69, jan 2019.

[27] *Sturdevant's Art and Science of Operative Dentistry*. Elsevier, 2019.

[28] J. E. Frencken, P. Sharma, L. Stenhouse, D. Green, D. Laverty, and T. Dietrich, “Global epidemiology of dental caries and severe periodontitis - a comprehensive review,” *Journal of Clinical Periodontology*, vol. 44, pp. S94–S105, mar 2017.

[29] *Dental Caries: The Disease and Its Clinical Management*. BLACKWELL PUBL, May 2015. [Online]. Available: https://www.ebook.de/de/product/23695989/dental_caries_the_disease_and_its_clinical_management.html

[30] “Dental X-rays.” [Online]. Available: https://my.clevelandclinic.org/health/articles/11199-dental-x-rays

[31] H. Strassler and M. Pitel, “Using fiber-optic transillumination as a diagnostic aid in dental practice,” *Compendium of continuing education in dentistry (Jamesburg, N.J. : 1995)*, vol. 35, pp. 80–8, Feb. 2014.

[32] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, sep 2009.

[33] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, “A comparative analysis of object detection metrics with a companion open-source toolkit,” *Electronics*, vol. 10, no. 3, p. 279, jan 2021.

[34] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[35] “cocodataset/cocoapi,” May 2022, original-date: 2015-01-25T20:26:39Z. [Online]. Available: https://github.com/cocodataset/cocoapi

[36] S. Jadon, “A survey of loss functions for semantic segmentation,” in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2020, pp. 1–7.

[37] “An overview of semantic image segmentation.” May 2018. [Online]. Available: https://www.jeremyjordan.me/semantic-segmentation/

[38] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[39] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” pp. 448–456, Jun. 2015. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html

[40] Y. Wu and K. He, “Group normalization,” 2018.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[42] Y. Li and T. Yang, “Word embedding for understanding natural language: A survey,” in *Guide to Big Data Applications*, S. Srinivasan, Ed. Cham: Springer International Publishing, 2018, pp. 83–104. [Online]. Available: https://doi.org/10.1007/978-3-319-53817-4_4

[43] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[44] A. Bochkovskiy, C. Wang, and H. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

[46] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *CoRR*, vol. abs/1905.11946, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[47] B. Baheti, S. Innani, S. Gajre, and S. Talbar, “Eff-UNet: A novel architecture for semantic segmentation in unstructured environment,” in *2020 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, jun 2020.

[48] H. Zhou, Z. Li, C. Ning, and J. Tang, “CAD: Scale invariant framework for real-time object detection,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, oct 2017.

[49] R. Solovyev, W. Wang, and T. Gabruseva, “Weighted boxes fusion: Ensembling boxes from different object detection models,” 2019.

[50] M. Prados-Privado, J. G. Villalón, C. H. Martínez-Martínez, C. Ivorra, and J. C. Prados-Frutos, “Dental caries diagnosis and detection using neural networks: A systematic review,” *Journal of Clinical Medicine*, vol. 9, no. 11, p. 3579, nov 2020.

[51] F. Casalegno, T. Newton, R. Daher, M. Abdelaziz, A. Lodi-Rizzini, F. Schürmann, I. Krejci, and H. Markram, “Caries detection with near-infrared transillumination using deep learning,” *Journal of Dental Research*, vol. 98, no. 11, pp. 1227–1233, aug 2019.

[52] F. Schwendicke, K. Elhennawy, S. Paris, P. Friebertshäuser, and J. Krois, “Deep learning for caries lesion detection in near-infrared light transillumination images: A pilot study,” *Journal of Dentistry*, vol. 92, p. 103260, jan 2020.

[53] K. Moutselos, E. Berdouses, C. Oulis, and I. Maglogiannis, “Recognizing occlusal caries in dental intraoral images using deep learning,” in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, jul 2019.

[54] M. Moran, M. Faria, G. Giraldi, L. Bastos, L. Oliveira, and A. Conci, “Classification of approximal caries in bitewing radiographs using convolutional neural networks,” *Sensors*, vol. 21, no. 15, p. 5192, jul 2021.

[55] A. G. Cantu, S. Gehrung, J. Krois, A. Chaurasia, J. G. Rossi, R. Gaudin, K. Elhennawy, and F. Schwendicke, “Detecting caries lesions of different radiographic extension on bitewings using deep learning,” *Journal of Dentistry*, vol. 100, p. 103425, sep 2020.

[56] I. S. Bayrakdar, K. Orhan, S. Akarsu, Özer Çelik, S. Atasoy, A. Pekince, Y. Yasa, E. Bilgir, H. Sağlam, A. F. Aslan, and A. Odabaş, “Deep-learning approach for caries detection and segmentation on dental bitewing radiographs,” *Oral Radiology*, nov 2021.

[57] Y.-C. Mao, T.-Y. Chen, H.-S. Chou, S.-Y. Lin, S.-Y. Liu, Y.-A. Chen, Y.-L. Liu, C.-A. Chen, Y.-C. Huang, S.-L. Chen, C.-W. Li, P. A. R. Abu, and W. Y. Chiang, “Caries and restoration detection using bitewing film based on transfer learning with cnns,” *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[58] L. Lian, T. Zhu, F. Zhu, and H. Zhu, “Deep learning for caries detection and classification,” *Diagnostics*, vol. 11, no. 9, p. 1672, sep 2021.

[59] J.-H. Lee, D.-H. Kim, S.-N. Jeong, and S.-H. Choi, “Detection and diagnosis of dental caries using a deep learning-based convolutional neural network algorithm,” *Journal of Dentistry*, vol. 77, pp. 106–111, oct 2018.

[60] W. Kuang and W. Ye, “A kernel-modified SVM based computer-aided diagnosis system in initial caries,” in *2008 Second International Symposium on Intelligent Information Technology Application*. IEEE, dec 2008.

[61] X. Zhang, X. Han, C. Li, X. Tang, H. Zhou, and L. Jiao, “Aerial image road extraction based on an improved generative adversarial network,” *Remote Sensing*, vol. 11, p. 930, Apr. 2019.

[62] Y. Bayraktar and E. Ayan, “Diagnosis of interproximal caries lesions with deep convolutional neural network in digital bitewing radiographs,” *Clinical Oral Investigations*, jun 2021.

[63] “Welcome to Python.org.” [Online]. Available: https://www.python.org/

[64] “Reengineering Facebook AI’s deep learning platforms for interoperability.” [Online]. Available: https://ai.facebook.com/blog/reengineering-facebook-ais-deep-learning-platforms-for-interoperability/

[65] “airctic/icevision,” May 2022, original-date: 2020-05-04T01:57:02Z. [Online]. Available: https://github.com/airctic/icevision

[66] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[67] O. Yadan, “Hydra - a framework for elegantly configuring complex applications,” Github, 2019. [Online]. Available: https://github.com/facebookresearch/hydra

[68] W. Falcon et al., “Pytorch lightning,” *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[69] L. Biewald, “Experiment tracking with weights and biases,” 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[70] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[71] D. P. E. Riba, D. Mishkin, “Kornia: an open source differentiable computer vision library for pytorch,” in *Winter Conference on Applications of Computer Vision*, 2020. [Online]. Available: https://arxiv.org/pdf/1910.02190.pdf

[72] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, “MMDetection: Open mmlab detection toolbox and benchmark,” *arXiv preprint arXiv:1906.07155*, 2019.

[73] B. E. Moore and J. J. Corso, “Fiftyone,” *GitHub. Note: https://github.com/voxel51/fiftyone*, 2020.

[74] G. J. et al., “ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements,” Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370

[75] F. Hutter, H. Hoos, and K. Leyton-Brown, “An efficient approach for assessing hyperparameter importance,” pp. 754–762, Jan. 2014. [Online]. Available: https://proceedings.mlr.press/v32/hutter14.html

[76] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” 2019.

Appendix B

Aditional results

Model	AP	$AP@.3$	$AP@.5$	$AP@.75$	$AP@.5_S$	$AP@.5_M$	$AP@.5_L$
YOLOv5-l6	0.463	0.869	0.841	0.442	0.697	0.887	0.974
EffDet-D4	0.297	0.82	0.735	0.164	0.552	0.838	0.815

Table B.1: Comparision of trained models on the train part of stage three dataset

Model	Precision	Recall	F-score	γ
FRCNN-R101	0.69	0.64	0.664	0.662
FRCNN-R50	0.623	0.68	0.65	0.489
YOLOv5-m6	0.671	0.6	0.634	0.273
YOLOv5-l6	0.621	0.64	0.63	0.219
EfficientDet-D4	0.621	0.59	0.605	0.216
RetinaNet-swint	0.661	0.63	0.645	0.24
RetinaNet-R50	0.674	0.6	0.635	0.41

Table B.2: Precision, recall, and F-score based on the confidence threshold γ for the models trained on stage four datset

Model	Precision	Recall	F-score	γ
FRCNN-R101	0.701	0.67	0.685	0.664
FRCNN-R50	0.679	0.7	0.689	0.663
YOLOv5-m6	0.672	0.69	0.681	0.238
YOLOv5-l6	0.609	0.62	0.615	0.114
EfficientDet-d4	0.648	0.62	0.634	0.183
RetinaNet-swint	0.714	0.67	0.691	0.401

Table B.3: Precision, recall, and F-score based on the confidence threshold. Models were trained on stage-five dataset

Model	AR	$AR@.5_{10}$	$AR@.5$	$AR@.75$	$AR@.5_S$	$AR@.5_M$	$AR@.5_L$
YOLOv5-l6	0.559	0.895	0.956	0.559	0.916	0.971	0.971
YOLOv5-m6	0.547	0.899	0.957	0.541	0.933	0.971	0.971
YOLOv5-s6	0.545	0.877	0.951	0.549	0.916	0.968	0.968
Effdet-D1	0.531	0.87	0.958	0.488	0.909	0.978	0.978
FRCNN-R50	0.475	0.867	0.883	0.445	0.832	0.899	0.899
FRCNN-R101	0.478	0.864	0.89	0.458	0.815	0.917	0.917
RetN-swint	0.508	0.89	0.946	0.468	0.902	0.967	0.967

Table B.4: Average recall of models trained by improved training protocol

Model	Precision	Recall	F-score	Confidence threshold
YOLOv5-l6	0.689	0.69	0.689	0.271
YOLOv5-m6	0.74	0.64	0.686	0.326
YOLOv5-s6	0.692	0.64	0.665	0.295
Effdet-D1	0.739	0.64	0.686	0.339
FRCNN-R50	0.719	0.66	0.688	0.759
FRCNN-R101	0.704	0.63	0.665	0.684
RetN-swint	0.681	0.69	0.685	0.367

Table B.5: Precision, recall, and F-score based on the confidence threshold for different models, trained by the imporved training protocol

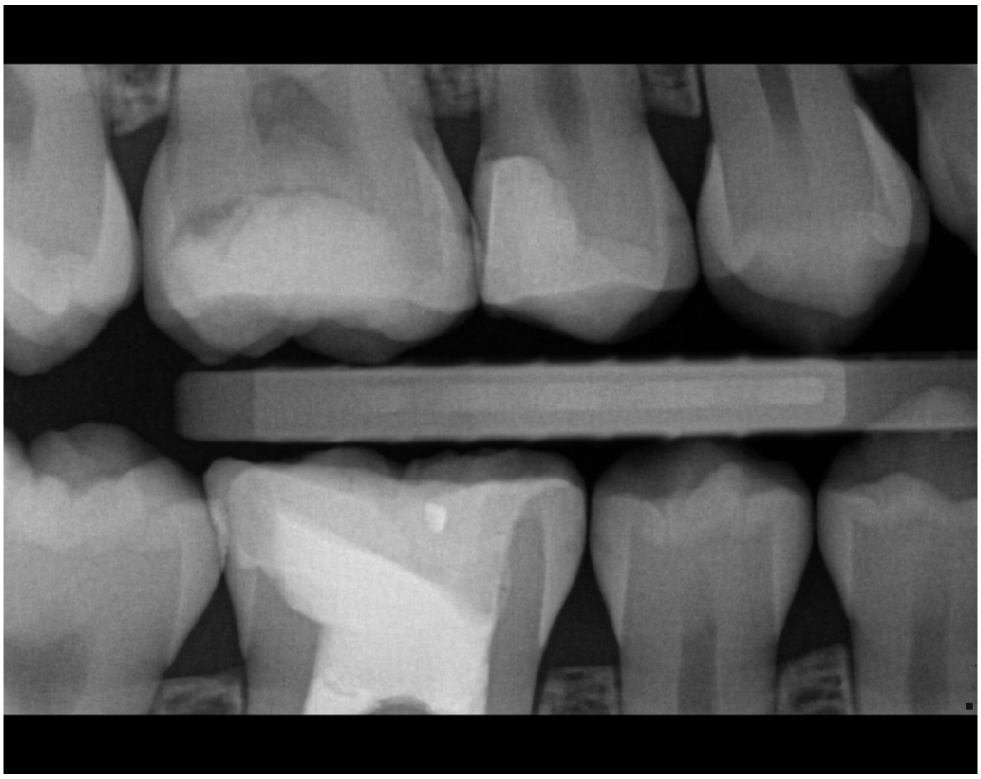
Method	AR	$AR@.5_{10}$	$AR@.5$	$AR@.75$	$AR@.5_S$	$AR@.5_M$	$AR@.5_L$
NMS	0.546	0.91	0.971	0.522	0.949	0.98	0.98
S-NMS	0.584	0.888	0.951	0.603	0.912	0.965	0.965
NWM	0.526	0.913	0.941	0.513	0.892	0.959	0.959
WBF	0.573	0.918	0.975	0.569	0.956	0.981	0.981
WBF-A	0.572	0.92	0.974	0.566	0.956	0.98	0.98

Table B.6: Average recall of models ensembled by parameters from Table 8.9

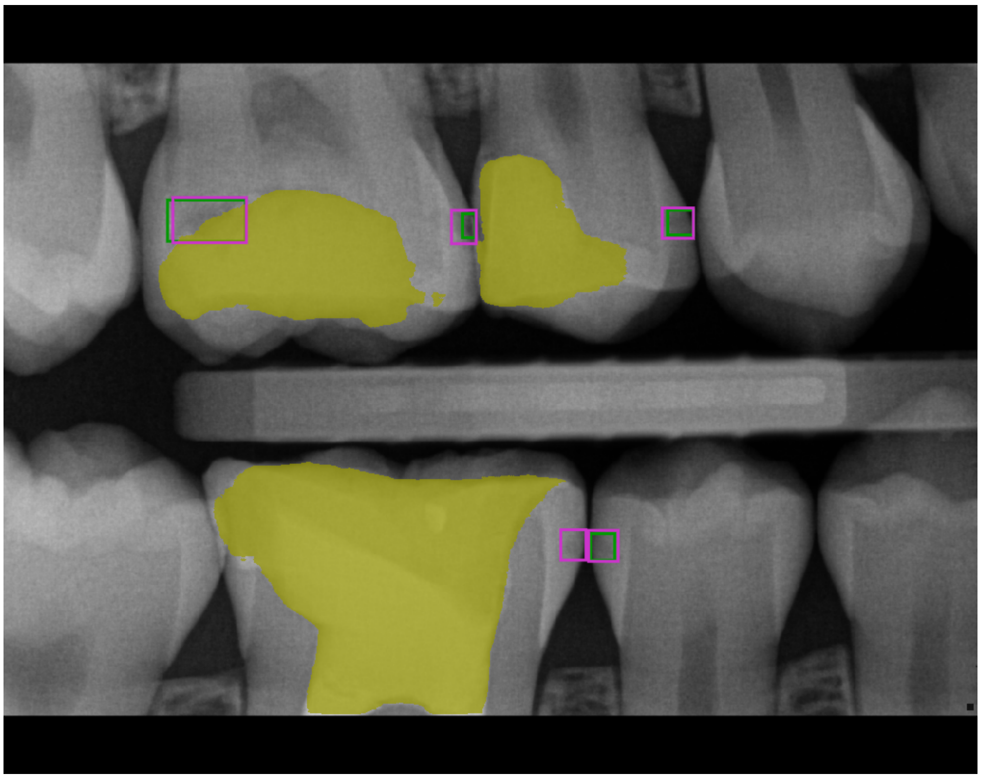
Method	Precision	Recall	F-score	Confidence threshold
NMS	0.708	0.68	0.694	0.284
S-NMS	0.679	0.7	0.689	0.489
NWM	0.713	0.71	0.712	0.792
WBF	0.732	0.7	0.715	0.594
WBF-A	0.745	0.69	0.716	0.201

Table B.7: Precision, recall, and F-score based on the confidence threshold for different ensembling methods

B.1 Detection of detnal caries and segmentation of dental restorations

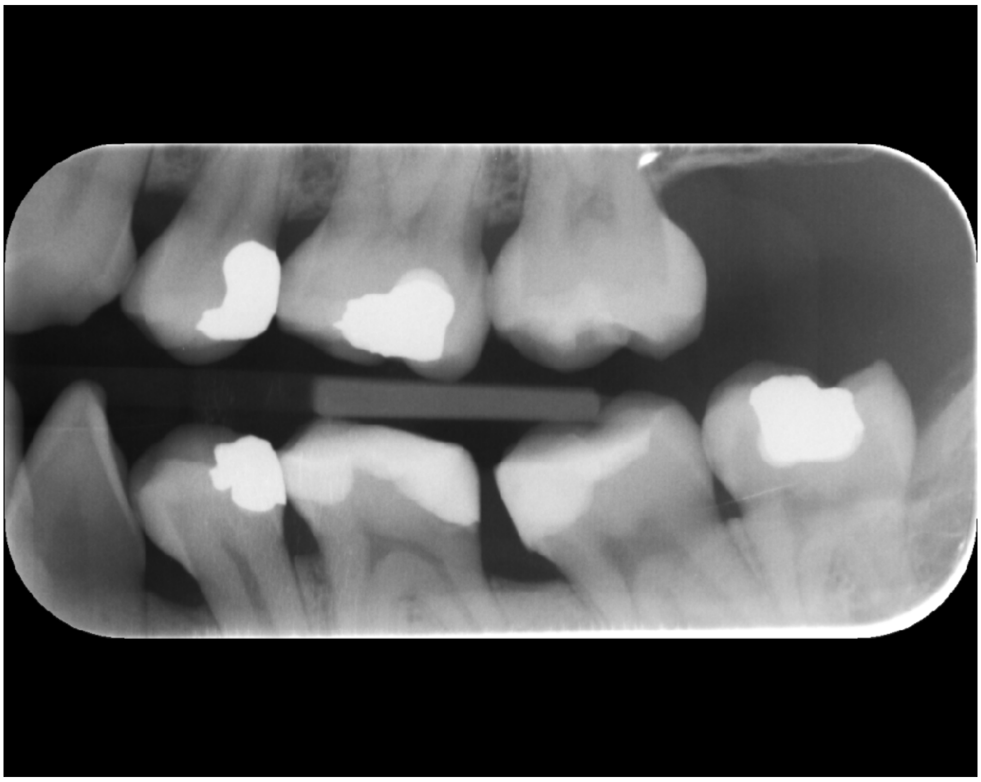


(a) : Input image

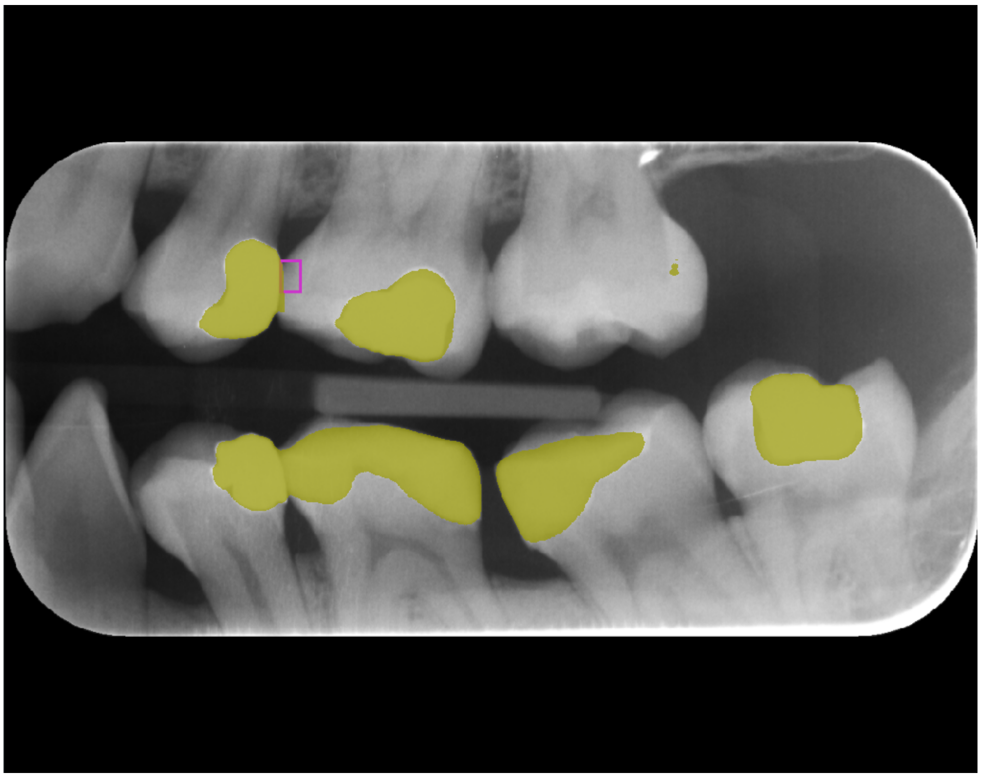


(b) : Predictions of the model

Figure B.1: Four correct detection of dental caries and one falsely-positive



(a) : Input image



(b) : Predictions of our model

Figure B.2: Segmented dental restorations in yellow, predicted dental caries are pink. We see a successful segmentation of dental restorations and a single falsely positive prediction of dental caries

Appendix C

Images

C.1 Predictions of the model

The following figures are X-ray images from the test part of the stage four dataset. Ground truth labels are marked by pink color and the model predictions are in green, please note the difference in used colors from Section B.1. Each bounding box prediction has corresponding confidence attached to it.

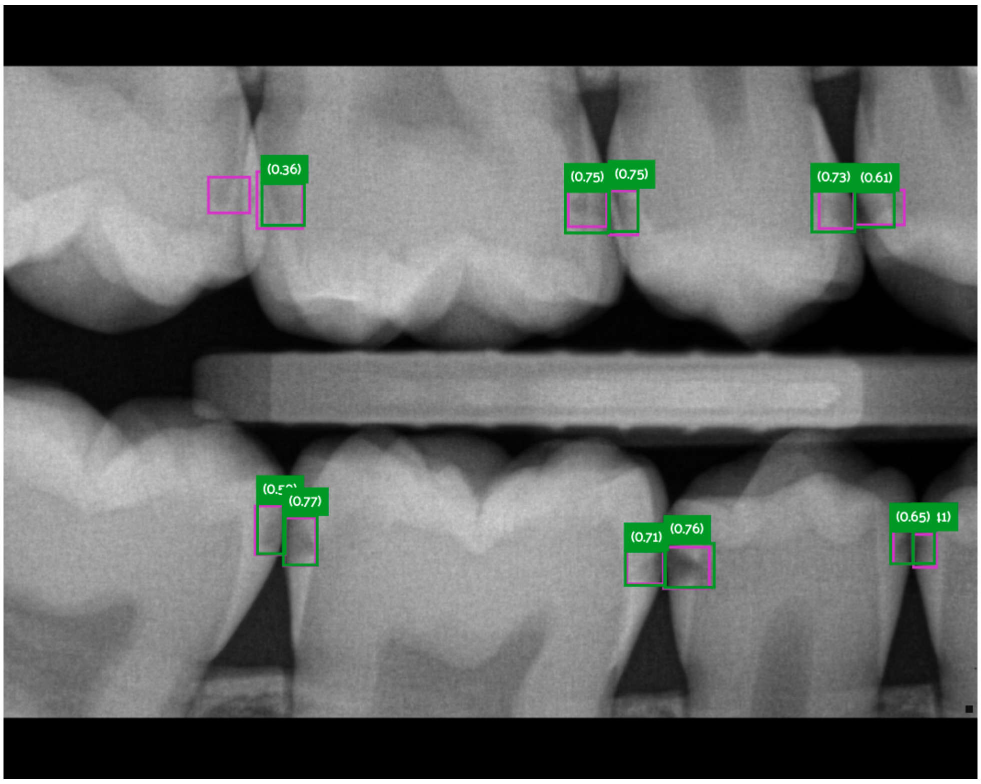


Figure C.1: X-ray image with ground truth boxes and model's predictions. Tho model correctly predicts 11 out of 12 dental caries in the image.

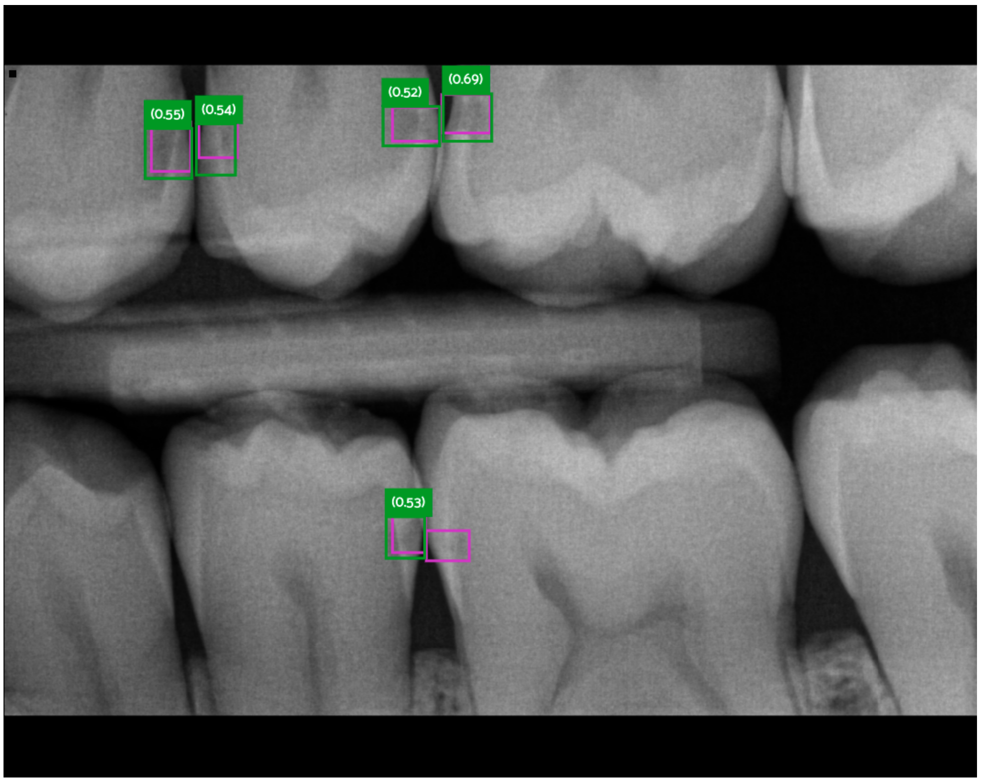


Figure C.2: The model predicts correctly five dental caries and has one falsely negative prediction

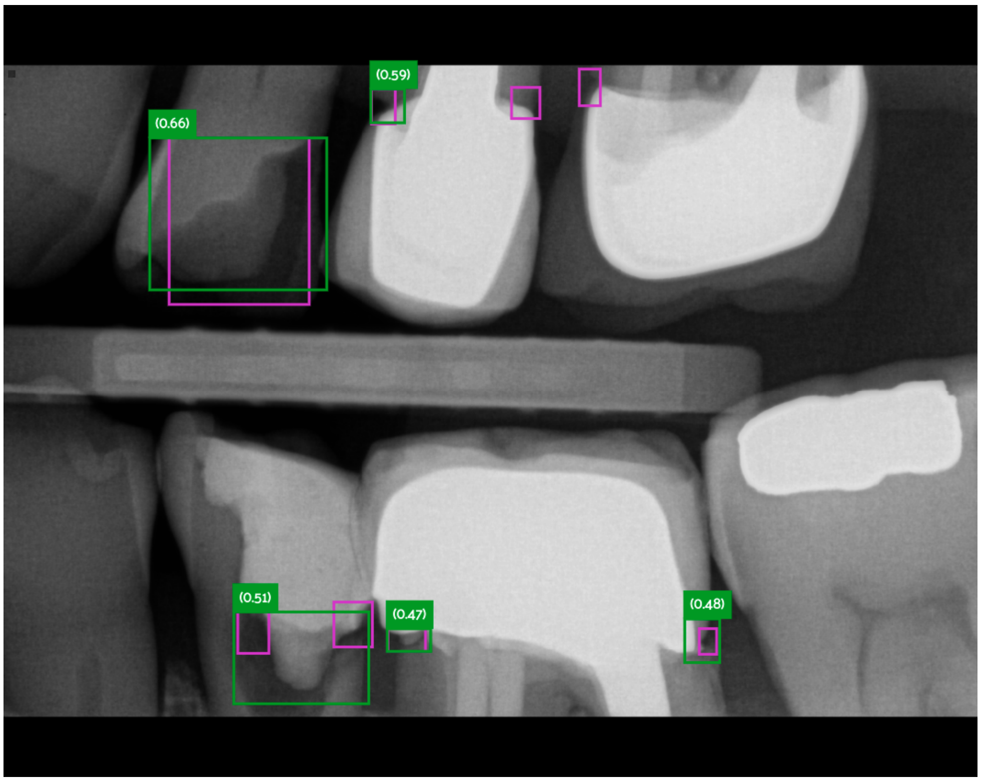


Figure C.3: Prediction of the model in image with extensive amount of dental restorations

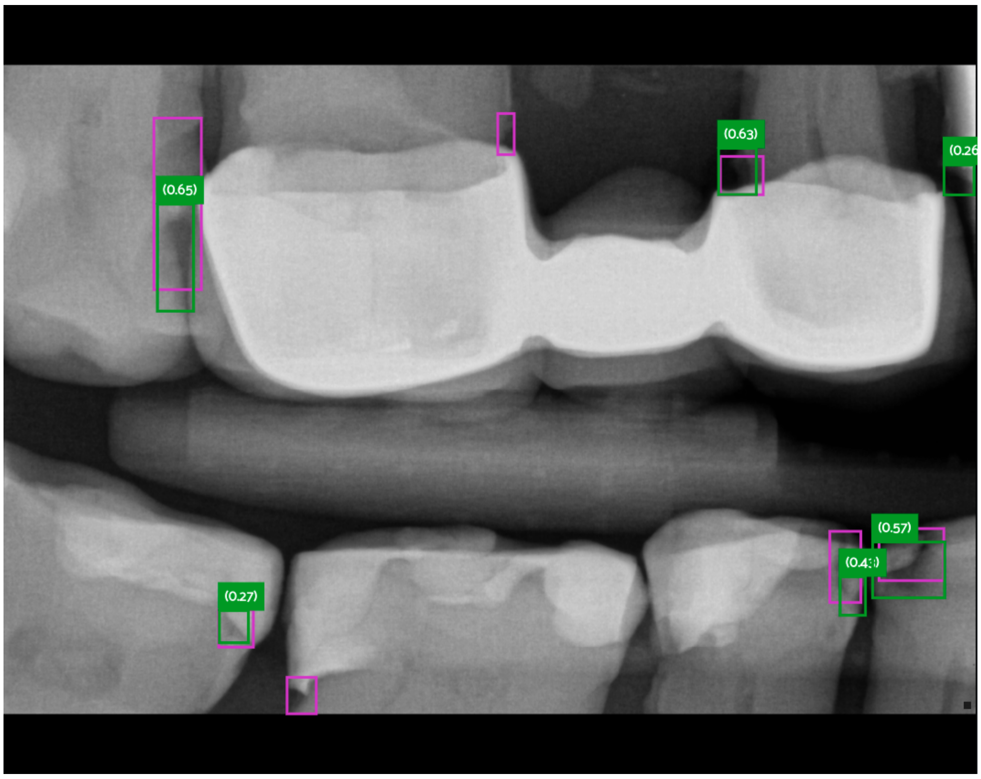


Figure C.4: The model predictions on image with dental bridge

C.2 Model importance during ensembling

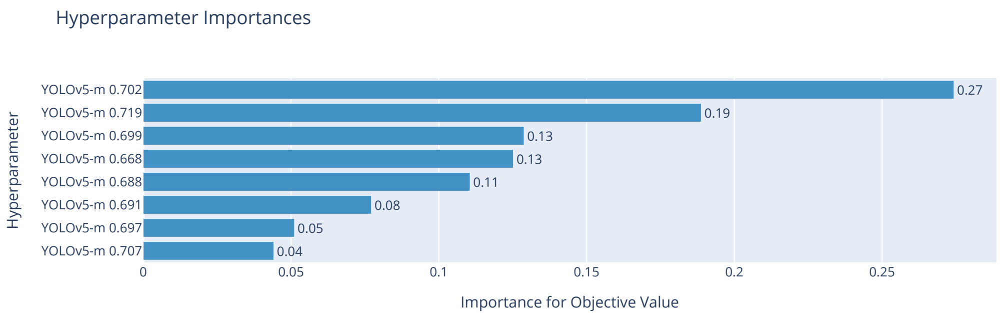


Figure C.5: Importance of different models during ensembling with the same architectures and backbones

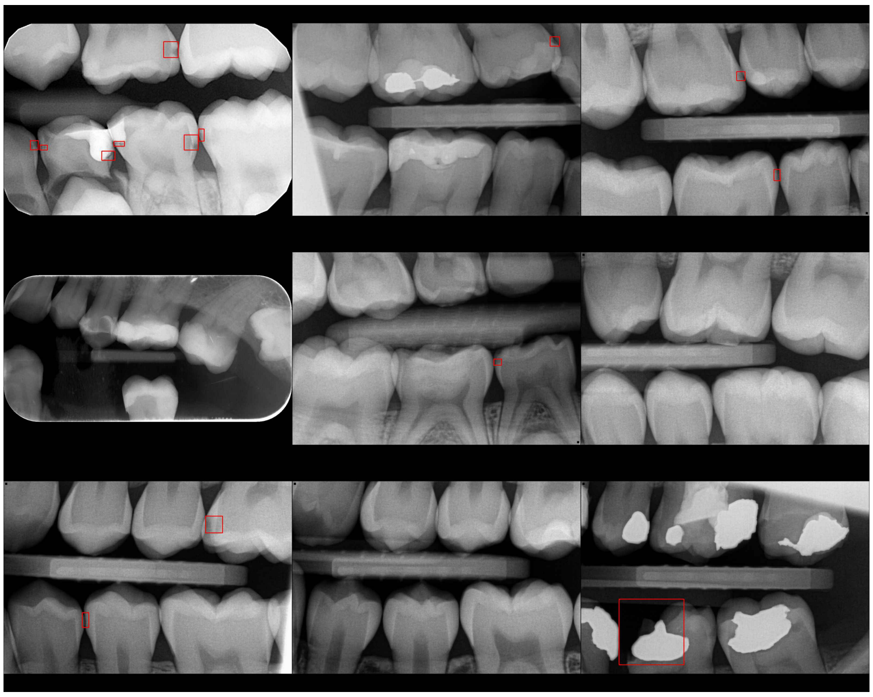


Figure C.7: No transformation applied

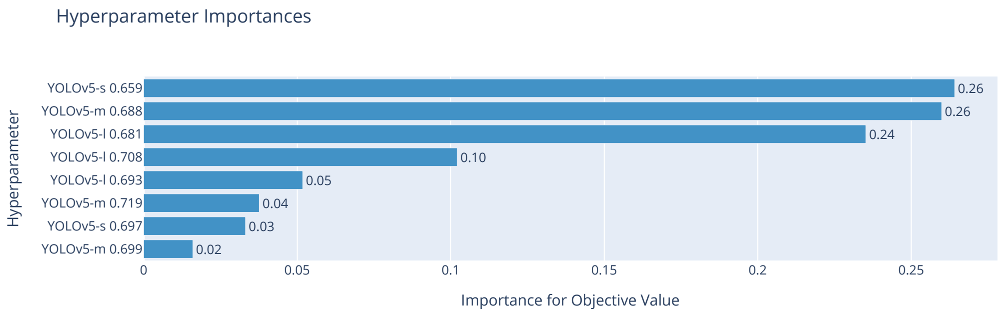


Figure C.6: Importance of different models during ensembling with the architecture and different backbones

C.3 Augmented images

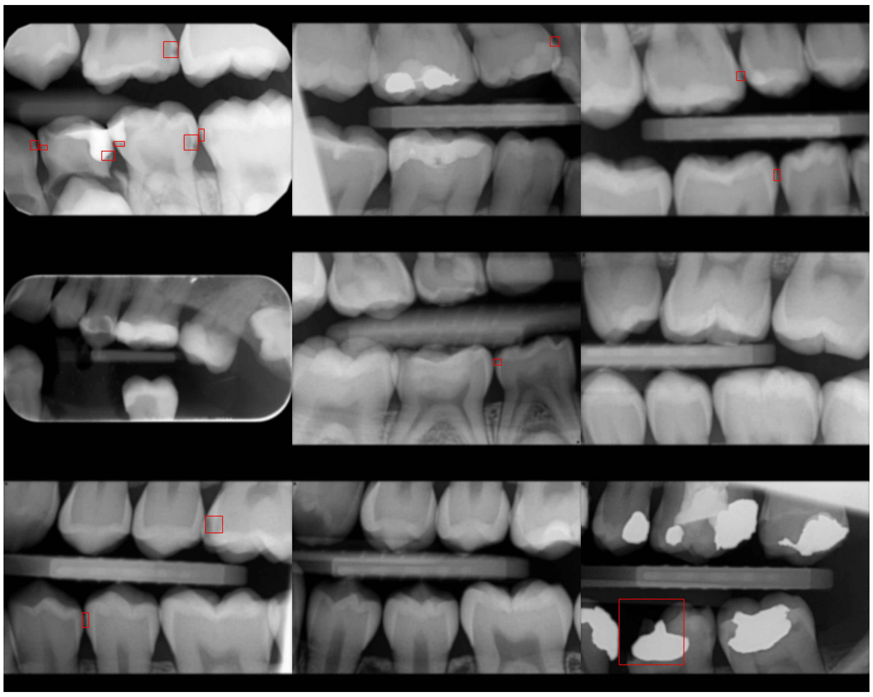


Figure C.8: Gaussian blur applied

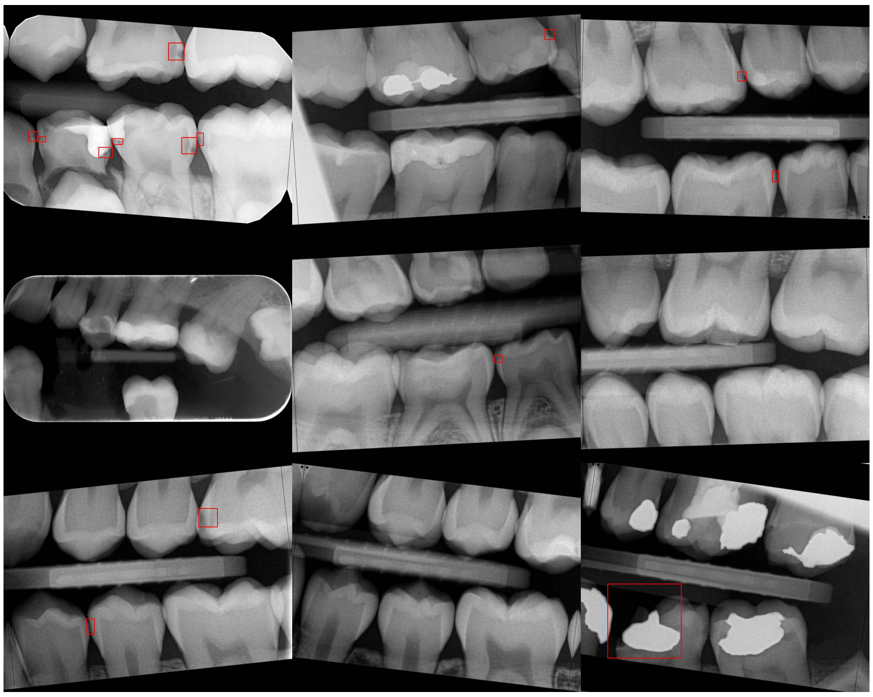


Figure C.9: Rotation applied

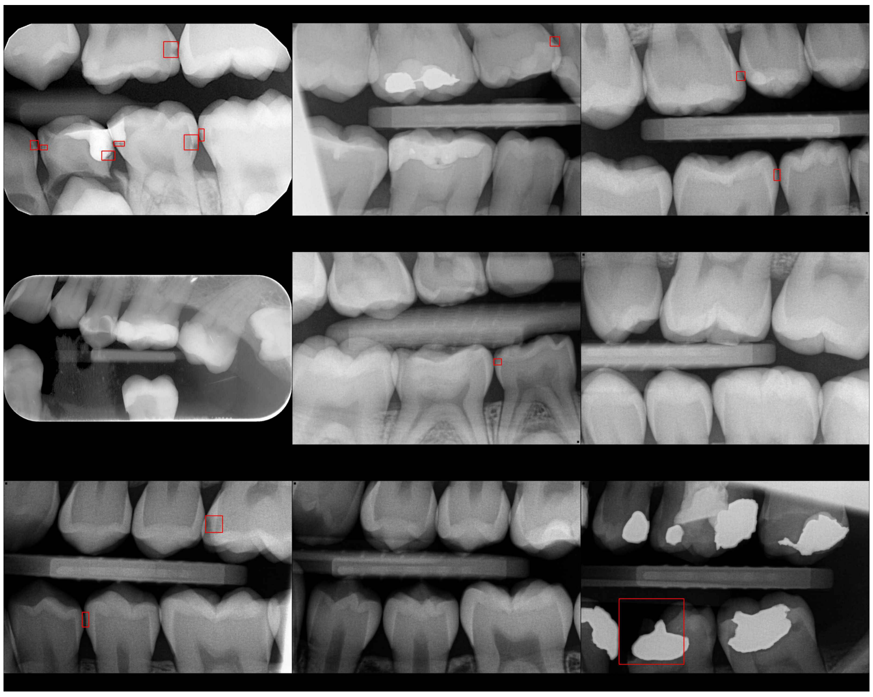


Figure C.10: Gamma correction applied

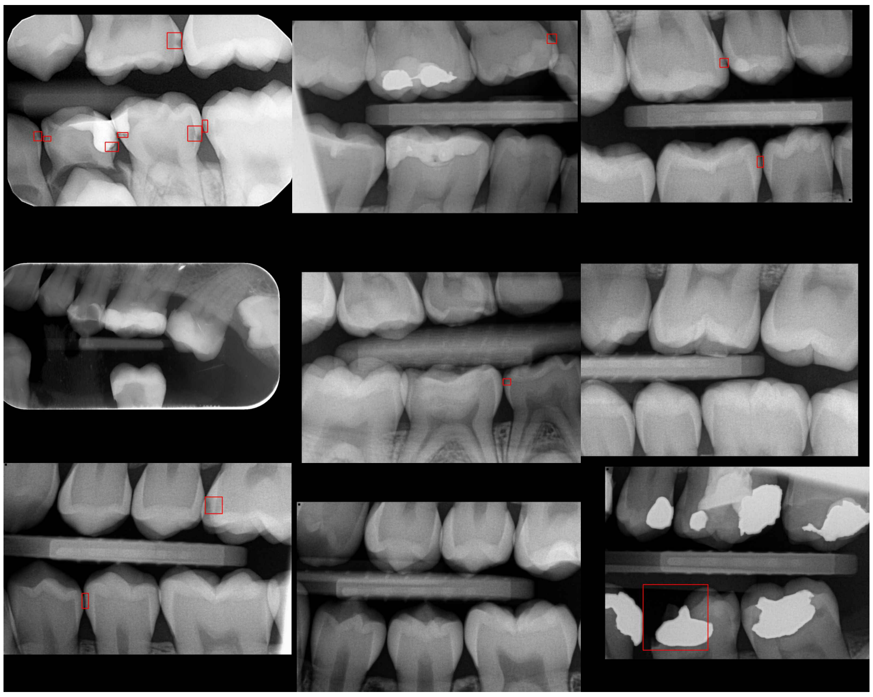


Figure C.11: Translation applied

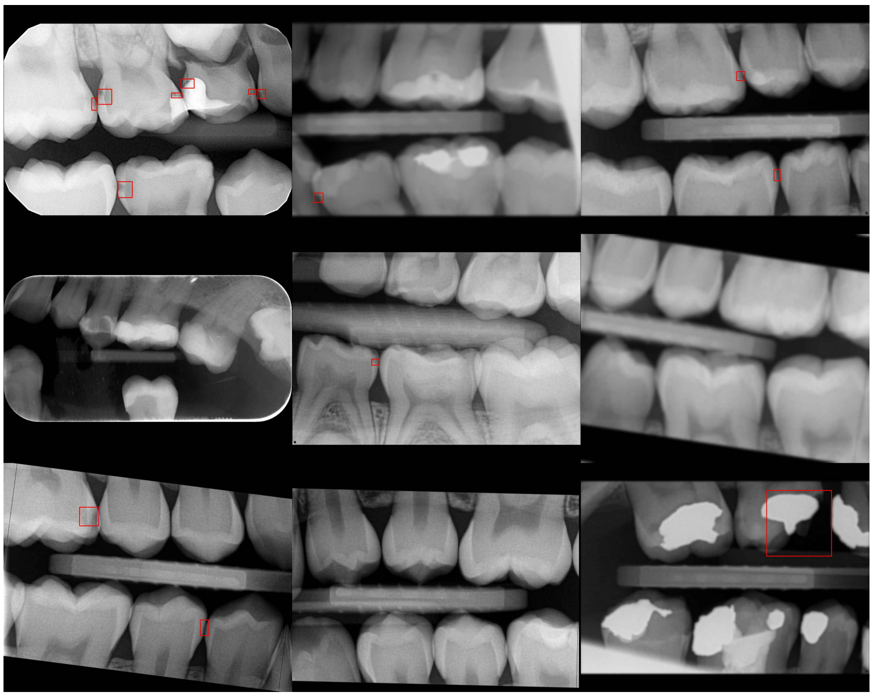


Figure C.12: The whole augmentation pipeline applied