DTU Compute
Department of Applied Mathematics and Computer Science

Context-Aware Object Detection
with use of deep learning

Master Thesis

Context-Aware Object Detection
with use of deep learning

Master Thesis
June, 2023

By
Spyridon Vlachospyros

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Brovej, Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk/
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis was written by Spyridon Vlachospyros in collaboration with Radiobotics ApS. The main advisor and supervisor of the project at Radiobotics were Jonas Christophersen who provided the author with guidance on the overall problem, data, and consultation. Additionally, the thesis was supervised by Nicki Skafte Detlefsen Postdoc, Section for Cognitive Systems, at the Technical University of Denmark (DTU).

The thesis was developed over a period of six months at the Department of Applied Mathematics and Computer Science, DTU, as part of Spyridon Vlachospyros' pursuit of the Master of Science in Human Centered Artificial Intelligence (MSc Eng) degree.

While reading this thesis, it is assumed that the reader possesses a foundational understanding of Statistics, Machine Learning, and Deep Learning.

The code implementation is publicly available in GitHub through this link: Thesis Repository

Spyridon Vlachospyros - s213160

S.Vlachospyros

Signature

22/06/2023

Date

Abstract

Fracture detection plays a crucial role in medical diagnostics, and deep learning models have shown promise in improving accuracy and efficiency in this domain. This thesis investigates the application of different deep learning architectures, namely Faster RCNN and Detection Transformer, for fracture detection, with a focus on pediatric wrist X-rays.

The Faster RCNN model is renowned for its exceptional performance in object detection tasks, including fracture detection. On the other hand, the Detection Transformer model introduces the concept of ”Context-Awareness” through its attention mechanism, forming a hypothesis that this type of model is more suitable for problems like Fracture detection. ”Context-Awareness” refers to the process of detecting and identifying objects in an image by considering the surrounding context and incorporating it into the detection algorithm to improve accuracy and understanding of the scene. Also, the value of study-wise prediction in a multiview approach to the problem is also investigated with the same models. The primary objective is to compare these two architectures and approaches, aiming to identify the most accurate and effective suggestion model to assist doctors and healthcare practitioners. The experimental and evaluation processes are designed accordingly.

Surprisingly, the initial assumption that the Detection Transformer model might outperform the Faster RCNN model is proven incorrect. The experimental results consistently demonstrate that the Faster RCNN model maintains its reputation as a robust detector, outperforming the Detection Transformer model in both single-view and multiview approaches. The Faster RCNN model achieves an impressive AP@50 value of 89.9%, further reinforcing its suitability for fracture detection. The multiview approach also shows promising results, with the Faster RCNN model achieving an AP@50 value of 87.8%. These findings highlight the importance of considering established models like Faster RCNN for fracture detection tasks. The study of multiview approaches also indicates the potential benefits of considering the entire study when making predictions.

These findings provide valuable insights for healthcare practitioners and pave the way for further advancements in fracture detection and medical imaging research. Future research can explore alternative architectures, datasets, and fine-tuning strategies to potentially leverage the strengths of the Detection Transformer model in fracture detection.

Acknowledgements

I would like to express my sincere gratitude to Radiobotics ApS for their invaluable support and collaboration throughout the course of this thesis. Special thanks go to Jonas Christophersen, who served as my main advisor and supervisor at Radiobotics. Jonas provided me with all the technical but also mental support through this demanding 6 months period. His expertise and insightful feedback were instrumental in shaping the direction of this work.

I would also like to extend my appreciation to Nicki Skafte Detlefsen, my supervisor from the Technical University of Denmark (DTU). Nicki's guidance and expertise in the field were instrumental in ensuring the quality of this thesis. His valuable input and constructive feedback at any time needed greatly enhanced the research process and the overall outcome.

I am sincerely grateful to the entire team at Radiobotics for their hospitality, support, encouragement, and collaborative spirit. Their dedication to advancing the field of medical imaging and their willingness to share their expertise and resources were instrumental in the successful completion of this thesis.

I am truly privileged to have had the opportunity to work with such talented individuals and organizations. Their contributions have undoubtedly played a significant role in the successful completion of this thesis, and I am grateful for their support and guidance throughout this journey.

Contents

1 Introduction and motivation

Object detection is a domain of computer vision that over the past few years has risen to the attention of both the academic and industry world. The main reason behind that was the tremendous progress in computer hardware and the use of GPUs in the field of AI which allowed more complex and more computationally heavy algorithms and datasets to be processed.

Nowadays more and more applications are using object detection algorithms either directly or as a precondition for increasingly difficult computer vision tasks like target tracking, event detection, behavior analysis, and scene semantic understanding. The purpose of an object detection algorithm is to identify the presence, position, and size of objects in an image or video, and to classify them into predefined categories [1].

The field of object detection is incredibly vast and encompasses many different applications. However, for the purposes of this thesis, we will be narrowing our focus to the medical application of object detection, with a specific focus on fracture detection on X-rays. This is an important area of study, as accurate and efficient fracture detection can have a significant impact on patients and the healthcare system. Accurate and efficient fracture detection in healthcare has significant implications. It enables faster diagnosis, leading to prompt treatment and improved patient outcomes. At this point is worth mentioning that the aim of these systems is not to automate the diagnosis but rather to aid doctors to improve their capabilities in diagnosis. Automated systems reduce manual workload, improving efficiency and minimizing errors. This cost-effective approach optimizes resource allocation and enhances accessibility to accurate diagnosis, particularly in underserved areas. Overall, automated fracture detection streamlines healthcare processes, improves patient care, and contributes to advancements in radiology and orthopedics. By focusing on this particular application of object detection, the hope of making a meaningful contribution to the field of medical imaging and computer vision is rising.

Bone fractures are a serious public health issue around the world affecting both the patient's life as well as the doctor's workload and the healthcare costs. Fractures are a significant burden on individuals, families, societies, and healthcare systems because they can cause job absenteeism, decreased productivity, disability, diminished quality of life, health loss, and high healthcare expenses. Indicatively research made by the global burden of disease in more than 200 counties showed that in 2019 there were more than 175 million new fractures registered to public hospitals with an average cost of 10,000 US dollars for a hip fracture that considers being one of the most difficult and time-consuming cases of a fracture [2], The same study also showed that the patella or ankle; the femur and fractures of the hand, wrist, or another distal part of the hand made up the top three anatomical sites in terms of the frequency of fractures.

What also needs to be considered is the time that a medical professional is devoting to serve and diagnose a patient with a fracture as well as the accuracy and the human error inside this procedure.

The time it takes for a doctor to identify a fracture can vary depending on several factors, such as the location and severity of the fracture, the diagnostic tools available, and the doctor's experience and expertise. Common indications and symptoms of a fracture, such as pain, edema, and deformity at the location of the injury, are generally recognized by doctors. To confirm the diagnosis, they might also do a physical examination and request

imaging tests like X-rays or MRI scans. A doctor can ask for multiple X-rays for a single patient ranging from 1-4 depending on how easy is for him to identify the fracture. It is a common phenomenon that the doctor will need at least two X-rays in order to inspect the hand from two different perspectives and validate whether or not a fracture exists. In rare cases, the doctor can have 3 or 4 and then will have access to all hand perspectives. The diagnosis of a fracture could be simple and quick in some circumstances, but it might also call for additional research or consulting with other medical experts. As a result, it is challenging to estimate the typical time it takes for a clinician to detect a fracture. However, depending on the situation, the majority of fractures can be found within a few hours to a few days.

Misdiagnosis is a well-documented problem in the medical field, as research has shown that medical professionals are prone to such errors. Factors such as the healthcare provider's specialization and the timing of the examination can contribute to an increased likelihood of human error. In a two-year period, a hospital in the United States recorded almost 6,000 patients who visited the emergency department with injuries. The results of the hospital's research showed that 1% of all visits resulted in a misdiagnosis of a fracture, while 3.1% of fractures were not identified during the initial visit [3]. These numbers may seem to be low but what needs to also be considered is that 86% of these misdiagnoses had severe consequences for the treatment of the patients. Another remarkable outcome of that research was that the peak time of the errors was during the night sifts between 8 pm ad 2 am. Other research that also took place in the US showed that 90% of the errors that radiologist do are false negatives.

Lastly, it is worth noting that the use of artificial intelligence (AI) can contribute to enhancing the fairness of medical treatments. By leveraging AI's improved diagnostic capabilities, underserved regions such as rural or economically disadvantaged areas, where hospitals may lack access to experienced radiologists, can benefit greatly.

1.1 Purpose and motivation of the thesis

As discussed in the introduction, there is an urgent need to provide doctors and hospitals with new technologies to reduce the occurrence of human errors and accelerate the identification of patient fractures. A deep learning fracture detection model could be a highly beneficial tool for the healthcare system. Its development is motivated by the desire to help doctors make better and faster decisions, while simultaneously lowering healthcare costs. Furthermore, underprivileged regions with limited personnel and small local hospitals could significantly benefit from such a system. Most human errors are happening because the decisions are taken by under-train doctors whose specialty and experience is different from the one that a radiologist has. Radiologists are limited in most hospitals and many times, especially in the emergency departments the decisions are taken by other doctors. The ultimate objective of a project like this would be to establish a foundation that could eventually lead to full automation of the fracture detection process.

1.1.1 The overall objective

The main objective of the thesis would be to find the best object detection model that can serve the purpose and motivation explained earlier. Various state-of-the-art and older deep learning architectures will undergo testing and analysis. The thesis will also explore whether a model that considers multiple X-ray images from the same patient can improve detection accuracy and achieve higher scores, compared to a model designed to only analyze a single image at a time. All of the final results will be compared to reach a conclusive summary. To achieve these objectives, a pediatric trauma dataset containing only wrist X-ray images will be utilized. Although the wrist-specific nature of the dataset

may impact the final results, the possibility of generalizing the findings, if more data from all the skeleton parts are added, remains a potential scenario since the same models can be used for all body parts.

1.1.2 Thesis structure

To ensure clarity and coherence in this thesis, a Background chapter will follow the introduction, examining existing methods, model architectures, and evaluation metrics. Furthermore, a detailed explanatory analysis of the dataset will precede the modeling and experimentation phase. The modeling and results chapter will be divided into two distinct approaches. The first approach involves using each X-ray image from the dataset as a unique input to the model, with a decision made per image. In contrast, the second approach aims to combine all available X-rays for a patient, resulting in a final decision for the patient instead of each image. These two modeling approaches will be analyzed and discussed separately in the relevant chapters, with a final section dedicated to comparing and contrasting them.

It is assumed that the reader has some basic knowledge of Machine Learning, Deep Learning, Statistics and Linear Algebra. The knowledge background that is provided will focus on Object Detection, Model Architectures and Metrics used for the thesis.

2 Background

This chapter will be dedicated to elucidating the various technological advancements and algorithm implementations employed in this thesis. It is assumed that the reader possesses a foundational understanding of machine learning, probabilities, and mathematics. Consequently, this chapter aims to furnish the reader with supplementary knowledge essential for comprehending the experiments and outcomes presented in the thesis. Going through the chapter structure, fracture detection with and without deep learning will be analyzed first to narrow down in the thesis-specific problem and analyze all the models and the metrics that will be used.

2.1 Fracture detection

A fracture is characterized as a partial or complete break in the bone. Depending on how the break and the injury are, fractures can be categorized as open or close. An open fracture is one that the bone has damaged the skin and usually, there is also a wound. A closed fracture is one that the skin is intact and can be further categorized by the doctors. A fracture can be caused usually by an injury or fall but fractures caused by diseases are also common. The first step in treating a fracture is to identify it. This can be challenging especially with the closed fractures [4]. The current procedure requires from the doctor to take the patient's health history including a description of how the injury happened followed by a physical exam. Depending on the initial results the doctors can then ask for more tests that include:

- X-ray: a diagnostic procedure that captures images of inside organs, bones, and tissues on film using invisible electromagnetic radiation beams.
- MRI scans: a type of test that creates precise images of inside body structures using powerful magnets, radiofrequency technology, and a computer.
- CT scans: This test uses X-rays and a computer to make detailed images of the body. A CT scan shows details of the bones, muscles, fat, and organs.

All these techniques can help doctors identify whether or not a fracture exists. Depending on the severity of the case and how difficult is for the doctor to identify the fracture one of these techniques usually follows the physical exam. There is a big difference in the complexity and the cost of these tests and for this reason, usually the X-ray is mostly used [4].

2.1.1 X-rays image analysis

X-rays are a form of electromagnetic wave categorized as radiation. They are utilized in medical imaging to capture internal images of the human body. These images portray various body parts in varying shades of black and white. This contrast is a result of different tissues absorbing differing amounts of radiation. Bones, containing calcium, absorb the highest levels of X-rays, hence appearing white. Conversely, soft tissues such as fat absorb less radiation, resulting in a gray appearance. Air, on the other hand, absorbs the least amount of X-rays, causing the lungs to appear black in the images [5].

2.2 Deep learning in Object detection

Going back 20 years from now in the beginning of researching object detection problems, due to the absence of advanced image representation techniques, a majority of the initial algorithms for object detection relied on manually crafted features. Some worth mentioning detectors back on that day were the Viola Jones Detectors in 2001 [6] and the HOG Detector in 2005 [7].

Regrettably, object detection faced a stagnation phase beyond 2010 as the effectiveness of manually engineered features reached its limit. Nevertheless, in 2012, there was a remarkable resurgence with the emergence of convolutional neural networks (CNNs) and their ability to learn intricate and meaningful feature representations from images. The breakthrough in object detection came in 2014 when the Regions with CNN features (RCNN) approach was proposed. Since the advent of deep learning, object detection has been categorized into two main approaches: ”two-stage detection” and ”one-stage detection” [8]. This can also be depicted by the growth in research papers during this period and until today shown in Figure 2.1.

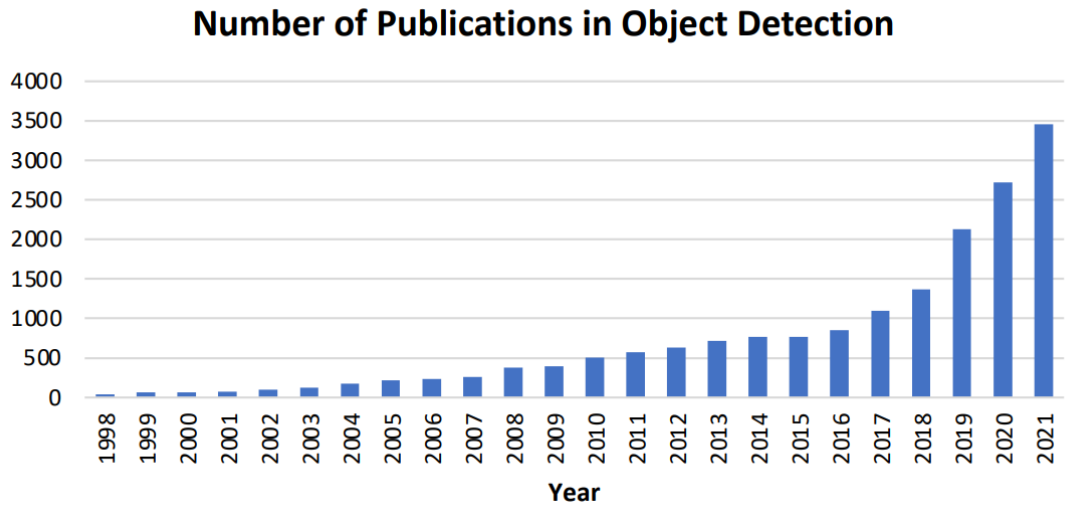


Figure 2.1: Number of publications in object detection from 1998 to 2021 [9]

The last big milestone for Object Detection with deep learning was when researchers start using transformer-based models that until then were used for Natural Language Processing (NLP). The field of deep learning, specifically computer vision, has experienced significant influence from Transformers in recent years. Transformers have revolutionized the conventional approach by replacing the traditional convolution operator with attention-based computations. This shift enables the overcoming of limitations in Convolutional Neural Networks (CNNs) and facilitates the capture of a comprehensive global-scale receptive field. In 2020, N. Carion et al introduced DETR [10], a novel perspective on object detection as a set prediction problem. They proposed an end-to-end detection network utilizing Transformers. Consequently, object detection has entered a new phase where the reliance on anchor boxes or anchor points is no longer necessary to detect objects [9].

2.2.1 Deep Learning Approaches

This chapter will examine various approaches employed in object detection, with a particular focus on the methods and techniques relevant to this thesis.

The different problems existing in object detection form different approaches, some of them can be applied in various projects while others are formed around specific applications. A general categorization is difficult, so the focus will be on the architecture of the models used. Two general categories that can classify all the models are ”two-stage detection” and ”one-stage detection”. The ”stages” are referring to the way the models are trained and the main difference is presented in the Figure 2.2.

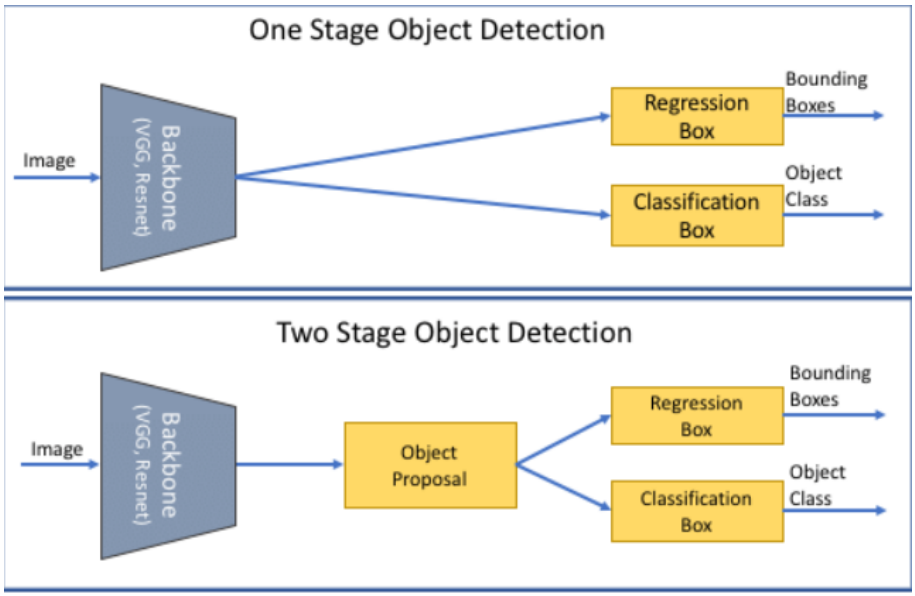


Figure 2.2: One versus two-stage architectures [11]

Two-stage detection refers to a type of object detection approach that involves two distinct stages. In the first stage, a region proposal network (RPN) is trained to propose different regions in the image that potentially include an object. These proposals serve as candidate regions of interest that can be used in the second stage. In the second stage, the proposed regions are classified and refined by a separate network that is trained separately in this task. Common techniques usually include region-based convolutional neural networks (R-CNN) or their variants. This two-stage process helps to improve accuracy by narrowing down the search space for objects and refining the detections.

One-stage detection, on the other hand, is an alternative approach where object detection is performed in a single step without the need for explicit region proposals. In this method, the network directly predicts the presence of objects and their corresponding bounding boxes at predefined spatial locations across the image. One-stage detectors, such as the DETR model and You Only Look Once (YOLO), typically use anchor boxes or default boxes at different scales and aspect ratios to handle variations in object sizes and shapes. One-stage detectors tend to be faster but may sacrifice some accuracy compared to two-stage detectors. Both techniques can have advantages and disadvantages depending on the application and one can outperform the other.

Other categorizations of the approaches used in real-life problems rely on the focus of the application. This means that based on the dataset used and the final goal different approaches are used and some of the most common ones in object detection are:

- 3D object detection is a computer vision task that involves recognizing and pinpointing objects within a three-dimensional setting, relying on their shape, position, and alignment. It entails detecting the existence of objects and accurately determining their spatial coordinates within the 3D environment in real time. This task holds significant importance in various applications, including autonomous vehicles, robotics, and augmented reality [12].
- Instance segmentation is the task of identifying different ”instances”, like individual people, in an image. Instance segmentation is very similar to object detection, except that we'd like to output a set of binary segmentation masks, rather than bounding boxes, with corresponding class labels. Instances are oftentimes also called ”objects” or ”things”. Similar to instance segmentation semantic and panoptic also exist [13].
- Real-Time Object Detection: Real-time object detection focuses on achieving high-speed object detection with minimal processing time, making it suitable for applications that require immediate and fast responses. Real-time object detection often involves optimizing the architecture and trade-offs between accuracy and efficiency to achieve high frame rates.
- Multi-Object Tracking: Multi-object tracking focuses on detecting and tracking multiple objects over time in video sequences. It involves associating object detections across frames to maintain object identities and trajectories. Multi-object tracking is essential in applications like surveillance, autonomous vehicles, and activity recognition [14].

In the following sections specific model architectures and approaches around object detection in X-rays and specific in fracture detection will be presented.

2.2.1.1 General field analysis in fracture Detection

Over the years the idea of making a decision-making support system for doctors and radiologists, especially for the emergency department was well established. There is a lot of research made by universities and other centers on fracture detection and some of them are going to be presented in this chapter to provide the reader with knowledge about related works.

Various studies have been conducted on fracture detection using artificial intelligence, utilizing both open-source and clinical bone image datasets from different medical devices. The area of interest in the human skeleton also varies among the studies but there are many examples with wrist datasets as the one that is going to be used for this thesis. Guan et al. achieved an average precision (AP50) score of 82.1% by employing a dilated convolutional feature pyramid network (DCFPN) on 3842 thigh fracture X-ray images [15]. Similarly, they obtained the highest AP result of 62.04% by employing a two-stage region-based convolutional neural network (R-CNN) method on around 4000 arm fracture X-ray images in the musculoskeletal radiograph (MURA) dataset [16].

Wang et al. achieved an AP50 score of 87.8% using the ParallelNet method, which utilized a TripleNet backbone network for fracture detection on a dataset of 3842 thigh fracture X-ray images [17]. Ma and Luo utilized Faster R-CNN for fracture detection on a subset of the dataset containing 1052 bone images. They further classified fractures using their proposed CrackNet model on the entire dataset, achieving an accuracy of 90.11% [18].

Sha et al. achieved a mAP score of 75.3% using the You Only Look Once (YOLOv2)-based model on a dataset of 5134 spinal fracture CT images [19]. They also utilized another Faster R-CNN model on the same dataset, achieving an mAP of 73.3% [20].

Finally, Xue et al. obtained an AP score of 70.7% using the proposed guided anchoring method Faster R-CNN model for fracture detection on 3067 hand fracture X-ray images [21].

In summary, the most commonly used models for fracture detection include dilated convolutional feature pyramid networks (DCFPN), region-based convolutional neural networks (R-CNN), and You Only Look Once (YOLOv2)-based models. It can be observed that the networks can differ despite the fact that the main goal is the same. The metrics are also varying among the models ranging from 73-88% in the AP50 value. Last but not least the authors of the dataset that is going to be used in this thesis also tried to apply a deep learning model on the dataset after they collected it achieving a high AP50 score of 0.933% with a pre-trained YOLOv5 model in a test set of 1000 images [22].

2.2.1.2 Multi-view vs single view approach of X-rays

The models presented in the previous chapter and most of the research around fracture detection usually use the same strategy. They approach the problem as a single view, meaning that they expect the models to have a single image as an input and the output of the model would be the predicted bounding boxes for this specific image. When a doctor request X-rays from a patient to detect whether or not their wrist has a fracture or not the number of X-rays per patient can vary. Depending on where the fracture is and how difficult it can be to detect it there can be one up to four different X-rays for the same wrist per patient. This can help the doctor have a panoramic view of the hand and detect a fracture that couldn't be visible from one side. An example of the pediatric wrist dataset that is going to be used for the thesis is presented below showing a case where the doctor took two X-rays from the wrist because only one fracture was visible in the left image 2.3.

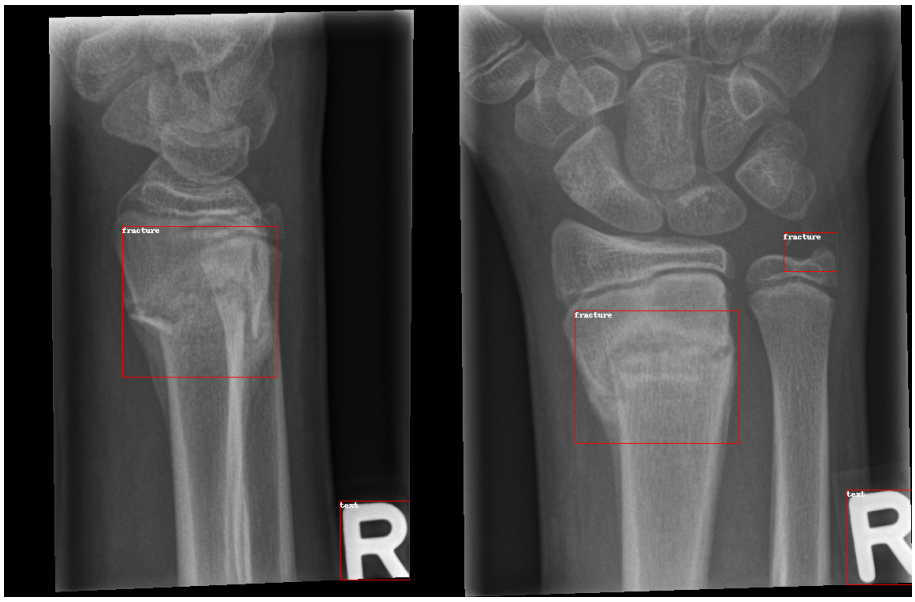


Figure 2.3: Example of two images for the same patient wrist

Similarly, following there is an example of a patient that needed four X-rays for the doctor to reassure that there was no fracture present Figure 2.4.

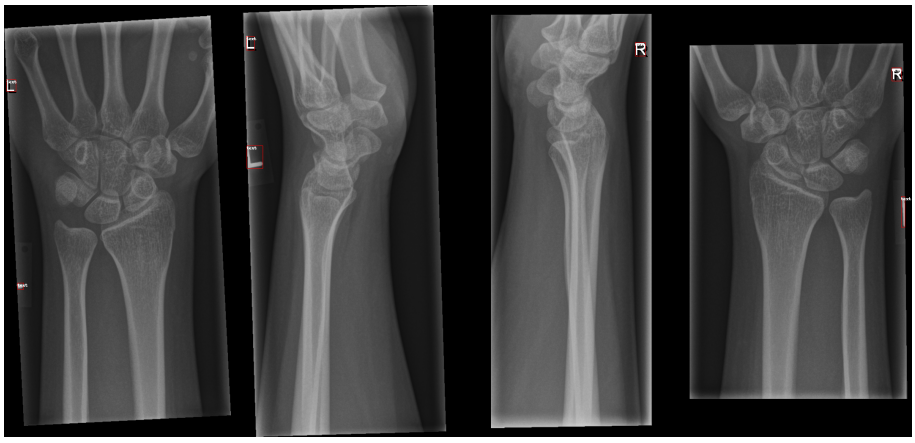


Figure 2.4: Example of four images for the same patient wrist

Four or three X-rays are rare to be seen per patient and that's because usually the doctor requires only one or two to decide. This aspect of the dataset is going to be analyzed

during the experiments and divided into two different approaches. In the first approach, a pipeline of one image input and prediction per image and not per patient will be followed like the ones that exist in most of the literature available. This approach is of course still valid since even if the patient has four different X-rays this pipeline can be used four times one for each image. Then depending on the application, the result can be presented in the doctor as four different outputs or as an ensemble method where the four results are combined into one based on a majority vote.

The second approach will focus on utilizing the aspect of the dataset that each patient can contain from one to four different X-rays. The idea behind this is that a deep learning model could take advantage of all the provided information per patient and predict the fractures from the collective information of all the images. This task is not easy to interpret since there is no pre-trained architecture of models accepting more than one image per instance. Some researchers dealing with similar problems are using 3D models mostly convolutional based to create 3D features of the object in the image [23, 24]. This strategy is not valid on the provided dataset and that's because most of the patients have one or two images and for the 3D layers to have value a full perspective of the hand is needed. So in order to provide all the information to the model at once, merging the images together and providing them as one image was chosen as the main strategy. The idea is that the model will be able to extract features from all the images and then based on that predict the bounding boxes more accurately providing at the same time the output to the doctor per patient and not per image. Another advantage of this approach is that all the known models and metrics can be used since the input at the end will be one collective image of all the X-rays. Both the images and the bounding boxes need to be preprocessed to support this approach.

2.3 Models

Going through the Models section, the reader will be presented with all the information about the models chosen to be used for the experiments later in the thesis. Analyzing in-depth other models' architecture is out of the scope of this thesis even if the names and some comparisons are made throughout the thesis. The models presented here are going to be used for both approaches with the same parameters and architectures so no separation is needed regarding this aspect.

First and foremost, two different models approaches and architectures were chosen to be used for the experiments. The first model is the Faster Region-based Convolutional Neural Network or F-RCNN proposed by Shaoqing Ren [25] in 2015. This model was a breakthrough in object detection when it was released and it's one of the most used models even today.

The second model is a transformer-based model called DEtection TRansformer or DETR first presented by Facebook AI in 2020 [10]. It's one of the state-of-the-art models used at the moment in object detection. All of the prior advancements, including the novel impact of transformers in computer vision, have been thoroughly explored in detail in the previous chapters. By leveraging self-attention mechanisms, transformers excel at capturing long-range dependencies in images, enabling them to model complex visual relationships and contextual information effectively. This capability has led to their successful application in tasks such as object detection, image classification, image generation, and semantic segmentation.

The reason behind the model choice is lying exactly in the reason that transformers tend to overcome CNN in most of the vision applications. The goal will be to compare one

architecture that has proven its efficiency and one new state-of-the-art architecture that can possibly outperform the legacy ones.

2.3.1 Models architecture

In this subsection, the models' unique architectures will be presented to the reader in-depth for a better understanding of the results and discussion section.

2.3.1.1 Faster-RCNN

The Faster-RCNN model is the latest and more advanced evolution of the RCNN model family. Its proposal in 2015 was revolutionary for object detection and until today it is one of the most used models due to its speed and accuracy. For a better understanding of the F-RCNN model, a small recap of the predecessors of this model is considered necessary.

The first model of this category was the R-CNNs (Region-based Convolutional Neural Networks) [26]. The basic model architecture is presented in Figure 2.5

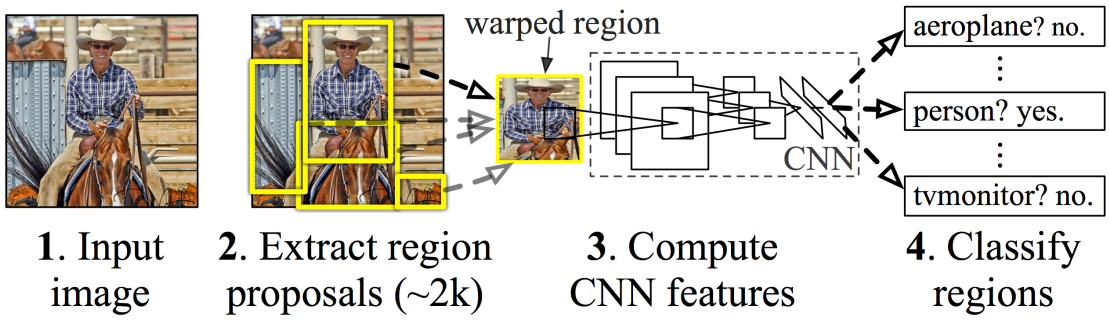


Figure 2.5: RCNN architecture proposed in 2014 [26]

The R-CNN model takes an input image and utilizes a technique known as the Selective Search to extract relevant information about the region of interest. The region of interest is identified using rectangle boundaries, and in certain cases, there can be a large number of such regions, potentially exceeding 2000. These identified regions of interest are then processed through a CNN (convolutional neural network) to generate output features. Subsequently, these output features undergo classification using an SVM (support vector machine) classifier, enabling the model to categorize the objects found within each region of interest. Another SVM network is also used at the end for the box regression problem [27, 28]. Although this model was also a breakthrough when it was released it has many issues making it impossible to be used in today's applications. Some examples of them are :

- Selective search is considered to be a time-consuming step for the training.
- Each image needs to classify 2000 or more region proposals. So, it takes a lot of time to train the network.
- It requires 49 seconds to detect the objects in an image on GPU.
- To store the feature map of the region proposal, lots of Disk space is also required.

To alleviate some of these issues the second model of the family was presented some months later called Fast-RCNN [29]. The architecture of this model is presented in Figure 2.6

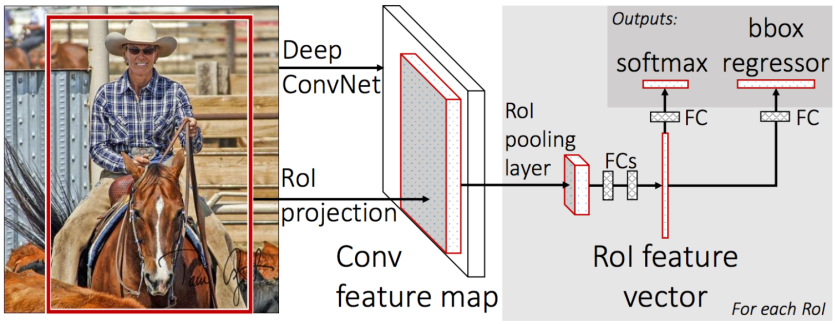


Figure 2.6: Fast-RCNN architecture proposed in 2015 [29]

The main difference that Fast-RCNN introduced was that algorithms like selective search were not included in the pipeline. In the new architecture instead of working on the 2000 region to convert them into feature maps, it converts the whole image on the feature map at once. This had as a result a significant difference in the computational times and the space the model needed while at the same time, it manage to have better results. All in all, Fast R-CNN performs feature extraction on the entire image using a CNN. Region of interest (RoI) pooling is then applied to the feature map, allowing for fixed-size feature maps to be extracted for each proposed region. These region-wise features are then fed into a fully connected network for classification and bounding box regression.

Finally, the architecture used in this thesis is the one of Faster-RCNN an updated version of Fast-RCNN [27]. This last proposed architecture of the RCNN family updated its predecessor with a region proposal network (RPN) to create the sets of regions. Faster R-CNN possesses an extra CNN for gaining the regional proposal, which is called Regional Proposal Network. In the training region, the proposal network takes the feature map as input and output region proposals. And these proposals go to the ROI pooling layer for further procedure as the architecture in Figure 2.7 depicts.

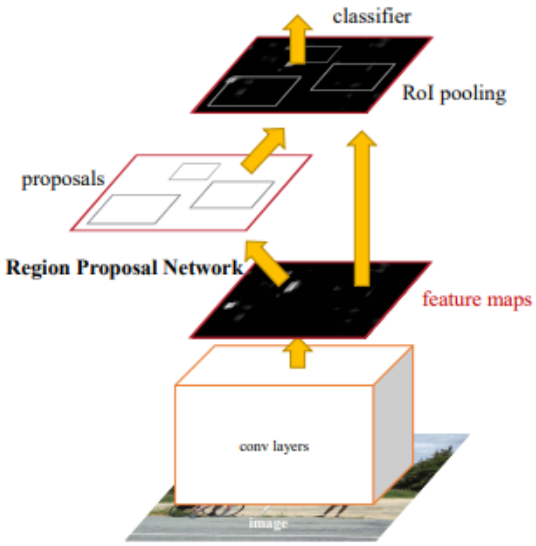


Figure 2.7: Faster-RCNN architecture proposed in 2015 [25]

The model improve dramatically the computational times in both training and predicting

allowing the researcher to use this model for real-time predictions.

The Faster-RCNN network belongs to two-stage detectors. This means that the network needs to be trained in two different steps. First, the RPN network needs to be trained, and then the rest of the network. A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score. To generate region proposals, a small network is applied to slide over the convolutional feature map produced by the final shared convolutional layer. This network takes an n × n spatial window from the input convolutional feature map as input. Each sliding window is then transformed into a lower-dimensional feature, such as 256-d for ZF or 512-d for VGG, using a ReLU activation function. These features are then passed through two sibling fully-connected layers: a box-regression layer (reg) and a box-classification layer (cls). While the images are passing through the RPN network anchors are used to help find all the objects present. Anchors are a set of predefined bounding boxes that serve as reference regions for object detection. Anchors are created at various scales and aspect ratios, covering different sizes and shapes of objects that are expected to be present in the image. These anchors act as reference templates for potential objects during the region proposal stage. 2.8 [25, 27]

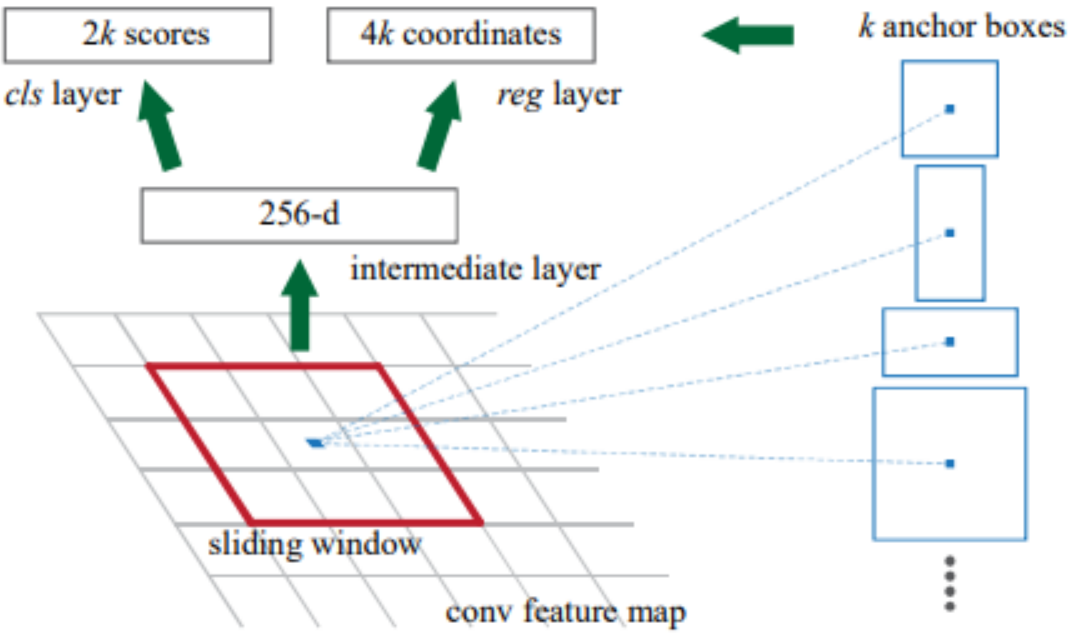


Figure 2.8: Region Proposal Network [25]

Concluding the model architecture a reference to the model loss is necessary and that's because the total model loss comes from the computation of two different losses. The loss from the RPN network and the loss from the Faster-RCNN model. The RPN is responsible for generating region proposals and consists of two subtasks: anchor classification and anchor regression. Each image is only classified into two classes either object or background. The total loss for the RPN network is calculated as:

$$\mathcal{L}\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{\text{cls}}}\sum_i \mathcal{L}_{\text{cls}}\left(p_i, p_i^*\right) + \frac{\lambda}{N_{\text{box}}}\sum_i p_i^* \cdot L_1^{\text{smooth}}\left(t_i - t_i^*\right)$$

Where $\mathcal{L}_{\text{cls}}$ is the logarithmic loss function across the two classes calculated as:

$$\mathcal{L}_{\text{cls}}\left(p_i, p_i^*\right) = -p_i^* \log p_i - \left(1 - p_i^*\right) \log\left(1 - p_i\right)$$

The loss function equation for the Fast R-CNN layer is quite similar but with two notable differences. Firstly, the samples are not obtained from the anchors; instead, they are extracted from the output of the region proposal layer, which consists of proposed bounding boxes. The determination of whether a sample is positive or not is based on the Intersection over Union (IoU) value between the proposed bounding boxes and the ground truth. Secondly, the classification loss for the Fast R-CNN layer is defined using a multi-class loss function, as illustrated below:

$$\mathcal{L}_{\text{cls}}\left(p_i, p_i^*\right) = -\log\left(p_i^c\right)$$

The final total loss used in Faster R-CNN is typically a linear combination of the RPN loss and the Fast R-CNN loss, with respective weights assigned to balance their importance during training. The mathematical expression is presented below:

$$\mathcal{L}_{\text{F-RCNN}} = \left(\lambda_{\text{F-RCNN}} * \mathcal{L}_{\text{F-RCNN}}\right) + \left(\lambda_{\text{RPN}} * \mathcal{L}_{\text{RPN}}\right)$$

Where λ_{FRCNN} and λ_{RPN} are the weights for the two losses and usually are set to 1.

The Fast R-CNN loss is applied to the region-wise features obtained from the RoI (Region of Interest) pooling layer. These features are fed into a fully connected network for classification and bounding box regression [25].

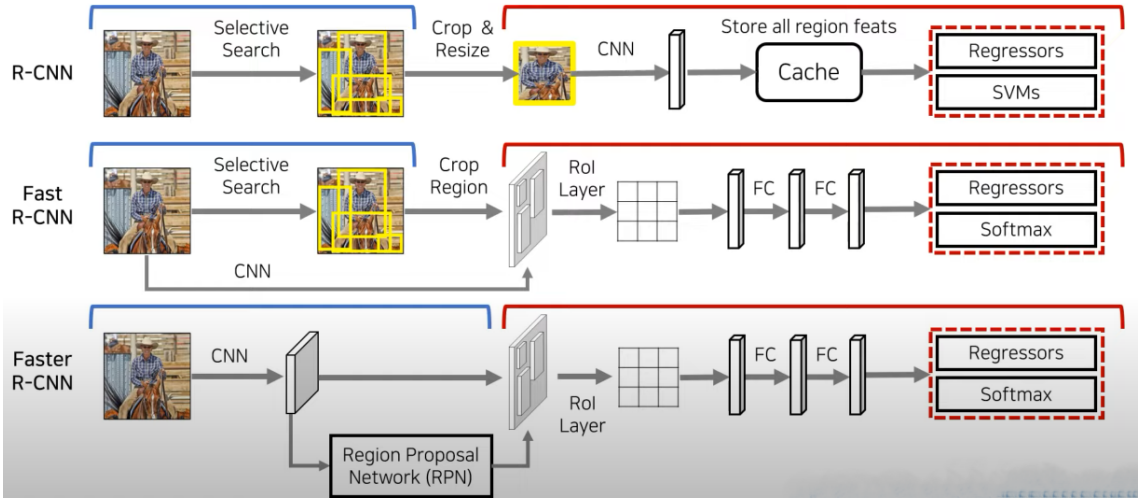


Figure 2.9: Comparison of RCNN architectures [30]

In this final Figure 2.9 all architecture of the RCNN family are presented with their differences for comparison. The big evolution is in the use of CNN layers in the feature extraction, at the region proposals, and in the final classification/regression problem.

2.3.1.2 Detection Transformer

Similar to the mind process followed for the Faster-RCNN before the architecture of the Detection Transformer is presented, a small introduction to transformers and especially vision transformers will take place. This will allow the reader to better understand this novel architecture.

The transformers model where first used with great success in different Natural Language Processing tasks especially after the publication of ”Attention is all you need” by Google researchers [31]. This groundbreaking paper marked a significant turning point in the field of Natural Language Processing (NLP). The authors introduced the concept of Transformer models, which swiftly gained widespread recognition for their remarkable performance across various NLP tasks. These Transformer models revolutionized the conventional approach to NLP by offering an alternative to recurrent neural networks (RNNs) and convolutional neural networks (CNNs), which had been widely utilized in the past. The paper advocated for the adoption of Transformer architectures, harnessing the power of attention mechanisms to capture long-range dependencies in sequential data. This paradigm shift not only yielded superior results but also substantially reduced training time. Since its publication, ”Attention Is All You Need” has become a cornerstone in the NLP community, sparking a new era of research and innovation in the field. This novel model architecture is presented below in Figure 2.10

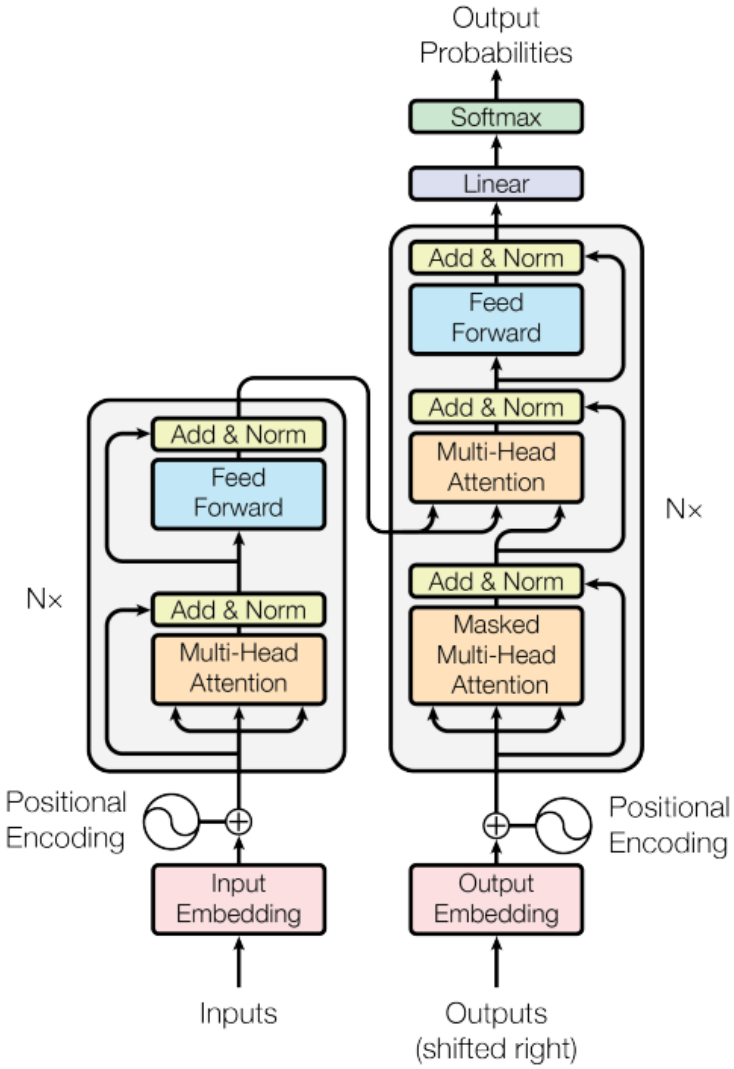


Figure 2.10: The original Transformer architecture proposed at [31]

The left part of the figure is presenting the Transformer Encoder and the right part the Transformer Decoder.

Encoder: The encoder consists of six layers that are all the same. Each layer contains two sub-layers. The first sub-layer is a multi-head self-attention mechanism, while the second sub-layer is a straightforward feed-forward network with connections between each sub-layer and the previous layer. Additionally, layer normalization is applied after each sub-layer, following a residual connection [31].

Decoder: The decoder is constructed with six identical layers, similar to the encoder. Each layer contains two sub-layers like the encoder, but it introduces a third sub-layer. This additional sub-layer performs multi-head attention over the output generated by the encoder stack. Like the encoder, the decoder incorporates residual connections around each sub-layer and applies layer normalization. In the decoder stack, the self-attention sub-layer is modified to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are shifted by one position, ensures that the predictions for a particular position, I, rely only on the known outputs at positions preceding I [31].

In addition to the two structural components mentioned, the attention mechanism in transformer models requires further elaboration, as it is the key element that sets these models apart and makes them unique. Unlike the typical approach in recurrent neural networks (RNNs) where attention is primarily directed toward the last encoder state, the decoder in this context takes into account all the states of the encoder. By doing so, it can access information about every element in the input sequence. This mechanism, known as attention, extracts relevant information from the entire sequence by calculating a weighted sum of past encoder states. Consequently, the decoder becomes capable of assigning higher weight or significance to specific input elements for each corresponding output element. By learning in each step, the decoder becomes adept at focusing on the appropriate input element to predict the subsequent output element [32]. The definition of attention as it is written in the original paper states:

> ”An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.” [31]

The previous explanation emphasized that attention scores are applied to the entire sentence simultaneously, resulting in identical outcomes even when two sentences have the same words arranged differently. However, it is desirable to direct attention to different word segments. To enhance the discriminative capability of self-attention, multiple self-attention heads are employed. The word vectors are divided into a fixed number of chunks (h, representing the number of heads), and self-attention is then applied to the corresponding chunks using Q, K, and V sub-matrices, as mentioned by Peter Bloem in ”Transformers from scratch” [33]. This process generates h distinct output matrices of scores.[32]

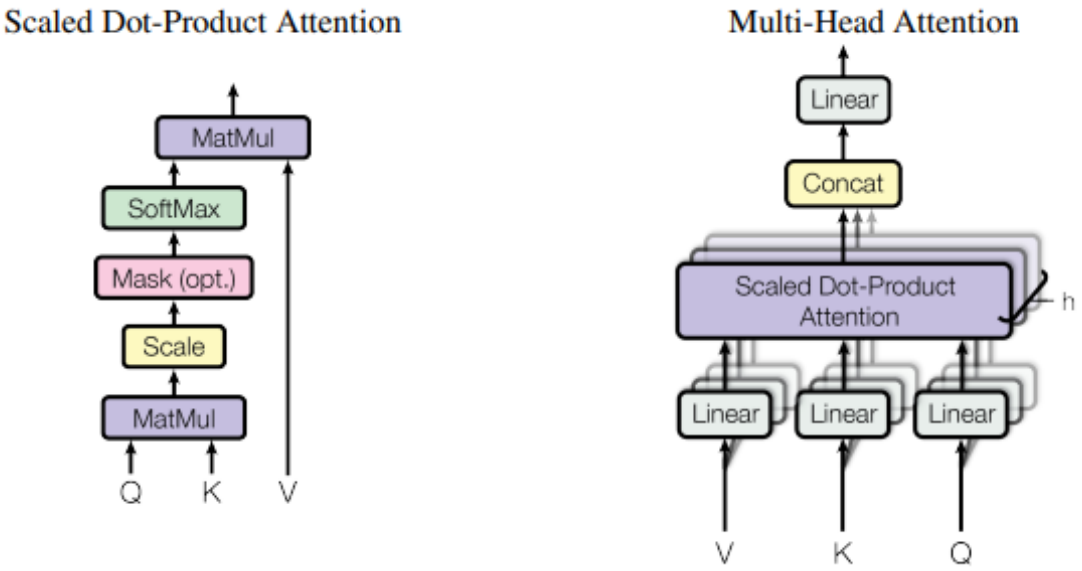


Figure 2.11: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel [31]

The attention is called in particular ”Scale Dot-Product Attention” Fig 2.11. The input consists of queries (Q) and keys (K) of dimension dk, and values (V) of dimension dv. The attention is calculated on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V [33].

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

Instead of using a single attention function with keys, values, and queries of the dimension model, it appeared that it is advantageous to linearly transform the queries, keys, and values h times. Each transformation employs distinct learned linear projections to dimensions dk, dk, and dv respectively. The attention function is then performed independently on each of these transformed versions, generating output values of dimension dv [33]. These values are concatenated and projected once again, producing the final values as illustrated in Figure 2.11.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right)W^O \\ \text{where head } &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)\end{aligned}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{madel} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{madel} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{modad} \times d_v}$ and $W^O \in \mathbb{R}^{hd_w \times d_\text{madel}}$.

What is more, the subsequent layer, known as the Feed-Forward layer, is designed to receive only a single matrix containing a vector for each word. In order to satisfy this requirement, the output matrices obtained from the previous step, after computing the dot product for each attention head, are concatenated. This concatenated matrix is then

multiplied by an additional weights matrix Wo, as described in [33]. The resulting final matrix effectively captures information from all the attention heads.

The innovative nature of this approach in Natural Language Processing (NLP), coupled with its superior performance compared to traditional architectures such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), has expanded its application potential. As a result, researchers in the field of computer vision became interested in exploring whether the same success could be replicated by applying a transformer model to image-based problems. A research paper named ”An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale” published in 2020 [34] proved exactly, that a transformer-based model could be made for NLP tasks and could have great success also in vision. The model architecture is presented below in figure 2.12

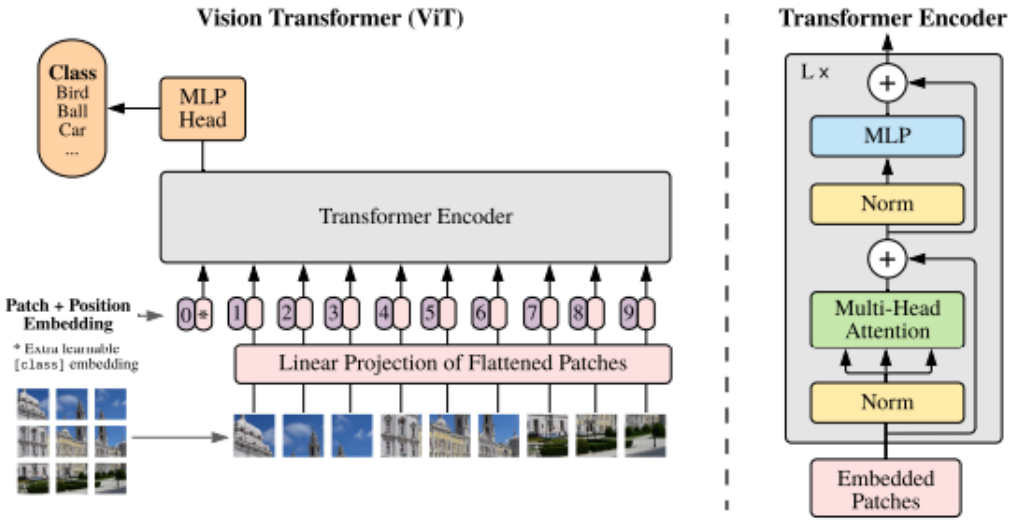


Figure 2.12: Vision transformer architecture [34]

Vision transformers adopt a unique approach to processing images. The image is divided into consistent, fixed-size patches, each of which is linearly transformed into an embedded vector representation. To preserve spatial information, position embeddings are introduced to indicate the relative positions of these patches. The sequence of embedded vectors is then passed through a conventional Transformer encoder. For classification purposes, an additional learnable ”classification token” is appended to the sequence, following the standard approach. By leveraging this innovative framework, vision transformers demonstrate their ability to handle visual data by effectively capturing both local and global dependencies [34].

All this research has led to advancements in various image-based tasks, including object detection, segmentation, and image classification. This was also the initial step for the implementation of the Detection Transformer in 2020 [10]. As it has been stated earlier there are two different approaches to solving an object detection problem. The most popular approach involves a two-stage architecture comprising classification and regression stages to identify target objects. In this method, region proposal generation is accomplished using techniques like Selective Search or Region Proposal Net (RPN) in the initial stage. Subsequently, classification and regression procedures are performed. Notable algorithms within this architecture include R-CNN, Fast R-CNN, and Faster R-CNN. While

these algorithms achieve high accuracy rates, especially for small objects, their speed does not meet expectations [35, 10]. The researcher of the DETR model took advantage of the expansion of the transformers and the lack of a reliable model that follows the second approach of a single architecture model. This approach focuses on completing object detection in a single stage, omitting the need for Selective Search or RPN. This method involves a single neural network model for the entire object detection process. While this approach prioritizes speed, there is a trade-off with accuracy, particularly for smaller objects [35, 10]. The model architecture to do that is presented in Figure 2.13 below :

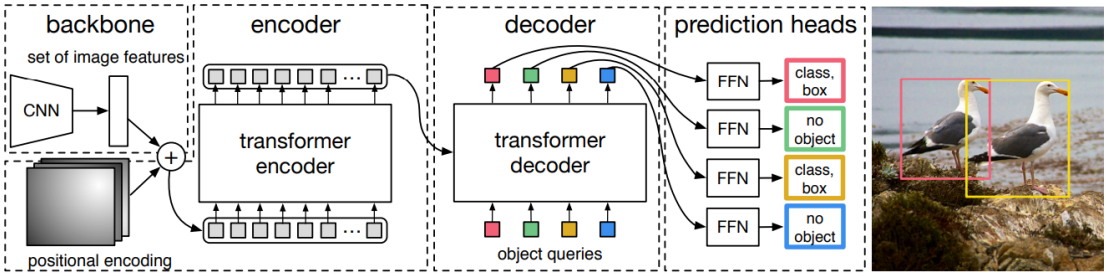


Figure 2.13: Detection Transformer Architecture [10]

The architecture of DETR, as illustrated in Figure 2.13, is remarkably straightforward. It comprises three key components, which will be described in detail below: a convolutional neural network (CNN) backbone responsible for extracting a concise feature representation, an encoder-decoder transformer that facilitates information integration, and a straightforward feed-forward network (FFN) that generates the ultimate detection prediction.

Backbone. The initial image x_{img} is processed by a conventional CNN backbone, resulting in a lower-resolution activation map f. The dimensions of x_{img} are $3 \times H_0 \times W_0$, representing three color channels. The generated activation map f has dimensions $C \times H \times W$, with typical values of $C = 2048$ and $H, W = \frac{H_0}{32}, \frac{W_0}{32}$ [10].

Transformer encoder. To begin, a 1×1 convolution operation is applied to reduce the channel dimension of the high-level activation map f from C to a smaller dimension d. This convolution produces a new feature map z_0 with dimensions $d \times H \times W$. Since the encoder requires a sequential input, the spatial dimensions of z_0 are collapsed into a single dimension, resulting in a feature map of size $d \times HW$. Each encoder layer consists of a multi-head self-attention module and a feed-forward network (FFN), following a standard architecture. In order to account for the permutation-invariant nature of the transformer architecture, fixed positional encodings are included as additional inputs to the attention layers [10].

Transformer decoder. The decoder follows the standard architecture of the transformer, processing N embeddings of size d using multi-headed self-attention and encoder-decoder attention mechanisms. However, unlike the original transformer, this model decodes N objects in parallel at each decoder layer. As the decoder is also permutation-invariant, the N input embeddings must be distinct to produce varied results. These input embeddings are learned positional encodings referred to as object queries, similar to the encoder. They are added to the input of each attention layer. The N object queries are transformed into an output embedding by the decoder. Subsequently, they are independently decoded into box coordinates and class labels using a feed-forward network (explained

in the next subsection), resulting in N final predictions. By utilizing self-attention and encoder-decoder attention across these embeddings, the model globally reasons about all objects together, considering their pairwise relations. Moreover, it can leverage the entire image as context for making predictions [10].

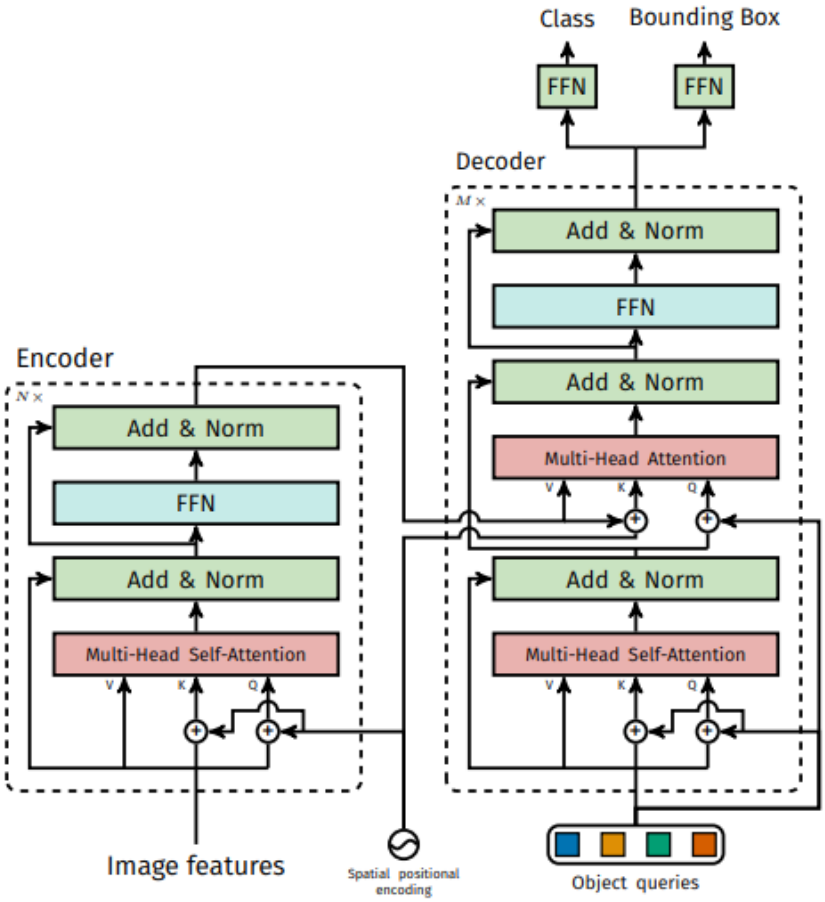


Figure 2.14: Encoder-Decoder detailed architecture [10]

In this detailed Encoder-Decoder architecture Figure 2.14, the reader can see all the common layers that are used for the Detection transformer with those proposed in the original transformed paper and presented in the Figure 2.10 before. The two architectures are almost identical proving that the Multihead attention used with great success in NLP tasks can also be applied to vision and especially object detection.

Trying to describe the whole architecture and how it operates, now that the structural pieces are explained, a step-by-step example is provided. In the initial stage, feature maps are extracted from the images using a Backbone layer, which can utilize various models like ResNet-50 or ResNet-101. This process helps preserve the 2-dimensional structural information. Subsequently, the data is flattened, converting it into a 1-dimensional structure. After applying positional encoding, the data is passed through the encoder-decoder mechanism. Finally, each output is forwarded to the Feed Forward Network. The last layer of the network comprises three nodes. By employing the ReLU activation function, the normalized center coordinates of the predicted object, as well as the predicted height and width values of the object, are obtained. The class of the corresponding object is predicted using the softmax activation function in another node. As a result, the need for

Non-Maximum Suppression (NMS) is eliminated [35, 10].

2.3.2 Architecture comparison and motivation

In the previous chapter, we have provided explanations for all the model architectures. In this section, we will briefly discuss the motivation and reasoning behind selecting these particular models. The primary objective of our model selection was to evaluate the performance of a novel architecture like DETR in fracture detection, an area where transformer architectures haven't yet proven their superiority, while Faster R-CNN architectures have demonstrated excellent performance. The approaches and implementations of these models differ significantly, with Faster R-CNN belonging to the category of two-stage architecture models, whereas DETR belongs to the one-stage architecture. Another reason for our selection is to examine the impact of global attention layers in the transformer model on the final results. Wrist fracture detection is a complex task due to the challenges in the model's ability to extract the same level of knowledge as a human doctor. Each individual has a unique hand and bone structure, and an abnormality in one person's bone may be considered normal in another. Therefore, doctors need to consider the entire hand structure to determine whether something is normal or abnormal. For instance, if all the fingers show an ”anomaly,” it might be considered ”normal” for a specific patient. Conversely, if only one of the fingers exhibits an abnormality, the doctor needs to take it into consideration. In these scenarios, a hypothesis is formed that the global attention mechanism of the detection transformer could potentially understand different types of anomalies, as the model is believed to have context awareness. This thesis aims to validate the value of this hypothesis and, more broadly, compare two different architectures to determine which one outperforms the other and why.

2.4 Transfer learning

This section will be devoted to a general introduction of transfer learning and its value to Deep Learning problems since it will be used later for the models analyzed before. Transfer learning is a technique used in Machine and Deep learning where a model trained and performing well in a specific problem is used as a starting point for another problem. A better definition is given by Goodfellow ”Transfer learning and domain adaptation refers to the situation where what has been learned in one setting … is exploited to improve generalization in another setting” [36] Trying to mimic the human way of learning new objective researchers find great value in this technique, especially in fields like computer vision and natural language processing given the vast computing and time resources required to develop a new neural network from scratch [37].

For computer vision tasks and especially medical image analysis problems, one of the most common issues is the lack of data mostly because in this section data are rare to find and expensive to label and are subject to GDPR rules [38]. Making a complex vision model to converge with limited data is quite difficult so transfer learning is a common technique used to initialize the models and then fine-tune them to the specific problem that needs to be solved.

Diving into X-ray-specific applications Transfer learning is also a common method used in different applications. Publications also showed that ImageNet pre-trained networks significantly improved performance across architectures on chest X-rays, with smaller architectures benefiting more from this boost [39].

ImageNet is a massive image database that acts as a standard in computer vision. The project has hand-annotated over 14 million photos to specify what items are seen, and bounding boxes are supplied in at least one million of the photographs while it has almost

20,000 categories [40]. ImageNet was founded to accelerate the development of computer vision algorithms, with a particular emphasis on picture classification problems. The database has been frequently used to train and test deep learning models, namely convolutional neural networks (CNNs), for tasks including object identification, detection, and localization. Because they capture a wide variety of visual ideas and attributes learned from a diverse set of pictures, pre-trained models on ImageNet have become a typical starting point for numerous computer vision applications. Some of the most commonly used networks that have been trained on image net are AlexNet, VGGNet, ResNEt, and EfficientNet all used today in different applications.

Following these trends, the model analyzed in the previous sections will be used in their pre-trained version since this option is applicable to both models.

Starting from the Faster RCNN model by Torchvison a pre-trained version of the model backbone region proposal network is offered [41]. The RPN network is a ResNet-50 trained on ImageNet. ResNet-50 is a specific variant of the Residual Network (ResNet) architecture. The rest of the network is pre-trained on the COCO object detection dataset offered by Microsoft. COCO is a large-scale object detection, segmentation, and captioning dataset and one of the most used datasets in object detection [42]. The idea behind this is to initialize the weight of the models from these pre-trained networks and then fine-tune the model in the X-ray images and the fracture detection problem needed to be solved in this thesis.

The DETR model from HugginFace is also offered with pre-trained weights from the same dataset as the FRCNN the COCO detection dataset [43]. The backbone network is also chosen to be a ResNet-50. This means that the initial settings of the model are the same and will allow us to fairly compare the models later.

2.5 Evaluation and Metrics

To conclude all the necessary knowledge needed to follow this thesis, the following section will focus on how an Object detection model is been evaluated and how these metrics enable researchers to compare different architectures.

As in every Machine/Deep learning problem collecting the data and fitting them to a model is just one piece of the puzzle. To ensure the reliability, robustness, and generalizability of these models, evaluating their performance becomes a crucial part of the entire process. Evaluating provides insides of the model's general accuracy and efficiency and allows researchers and stakeholders to take decisions and communicate their stories. That's why it's very important for the correct metrics to be chosen, the ones that can really depict the model's actual performance.

Object detection poses a unique challenge compared to other problems as it requires models to accurately predict both the class of objects and their corresponding locations. This complexity gives rise to a challenging evaluation task where the model's accuracy in each class and its ability to accurately localize objects must be considered together in the final metric. The evaluation process for object detection demands a comprehensive and cumulative assessment of the model's performance in detecting objects of different classes and precisely determining their spatial positions. All in all, the final metric is a combination of a classification and a regression problem.

2.5.1 Evaluation in Object detection

As it has been stated before each object detection dataset provides an image and bounding boxes with labels around each object present in the image. The output of a trained

model per image input is a collection of the model's predicted bounding boxes and their label for the image. The perfect model is the one that will manage to predict the exact spot and label of all objects in the image [44].

Because of the problem's complexity, some initial definitions of different methods and algorithms used in evaluating an object detection model will be provided here.

2.5.1.1 Intersection Over Union

One of the first things that need to be defined is the Bounding box similarity. When two boxes are considered to be similar and how this similarity is measured. Since having a perfect match between the ground truth box and the predicted one is rare to impossible to happen we need a metric that quantitatively calculates the boxes' overlap. In object detection to tackle this issue Intersection over union is used presented in Fig 2.15.

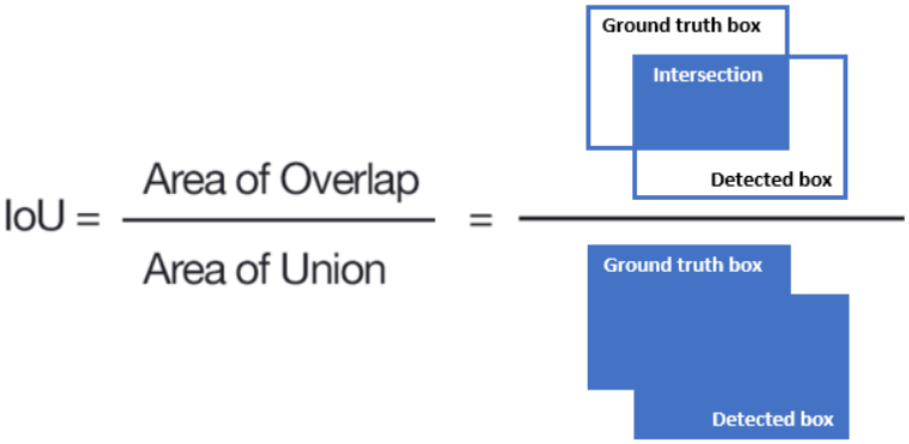


Figure 2.15: Intersection over union example [45]

Intersection Over Union (IOU) serves as a metric derived from the Jaccard Similarity, which assesses the degree of overlap between two bounding boxes. To calculate it both a ground truth bounding box and a predicted bounding box is needed. IoU is calculated by dividing the area of overlap between the predicted bounding box and the ground truth bounding box by the area of their union [46]. IoU is a value ranging from 0 to 1. If IoU is 1, the prediction perfectly matches the ground truth. If IoU is 0, there is no overlap. In order to accept a bounding box as valid we set an IoU threshold. If the IoU is equal to or greater than the threshold then it considers to be acceptable if not it counts as a false positive. Usually, a value between 0.5-0.7 is chosen as the threshold for the IoU but this can change depending on the application. The value of the threshold can really affect all the metrics that are going to be presented later and is considered to be one of the most important parameters of evaluating an Object detection model [45].

2.5.1.2 Precision and Recall

Two metrics that are commonly used among different Machine Learning problems have also value in object detection. Precision and recall are more accurate and usually preferred to simpler metrics like Accuracy.

Precision is the probability of the predicted bounding boxes matching actual ground truth boxes, also referred to as the positive predictive value. It measures the proportion of true positive predictions (correctly identified positive instances) out of all positive predictions made by the model (true positives plus false positives).

In other words, precision focuses on the quality of positive predictions, indicating how well the model avoids false positives [47].

$$Precision = \frac{TP}{TP + FP} = \frac{true\ object\ detection}{all\ detected\ boxes}$$

A high precision value indicates that the model has a low rate of falsely labeling negative instances as positive, suggesting a greater level of confidence in the positive predictions made by the model. Precision scores range from 0 to 1 [48].

The recall is the true positive rate, also referred to as sensitivity, and measures the probability of ground truth objects being correctly detected. It measures the ability of a model to correctly identify all positive instances out of the total actual positive instances in the dataset. In other words, recall evaluates the completeness of the model's positive predictions. It represents the model's capability to avoid false negatives, which are positive instances that were mistakenly classified as negative [47].

$$Recall = \frac{TP}{TP + FN} = \frac{true\ object\ detection}{all\ ground\ truth\ boxes}$$

A high recall value indicates that the model has a lower chance of missing positive instances, implying a better ability to capture all relevant information. However, a high recall may come at the cost of increased false positives. Therefore, it is often necessary to balance recall with precision, depending on the specific requirements and priorities of the problem at hand. Similarly, recall ranges from 0 to 1 where a high recall score means that most ground truth objects were detected[48].

2.5.1.3 Average Precision

It can be easily interpreted from the previous section that neither recall nor precision can be used as a single evaluation method. In a multiclass classification scenario, the model provides the probability that a bounding box corresponds to a particular class. A higher probability suggests a higher likelihood that the bounding box contains that specific class. By utilizing a probability distribution and a user-defined threshold (ranging from 0 to 1), the model categorizes the bounding box.

A lower probability confidence threshold leads to a greater number of detections by the model. This may enhance recall by reducing the chances of missing ground-truth labels (although not always). Conversely, a higher confidence threshold indicates a higher level of confidence in the model's predictions, resulting in improved precision (although not always). Ideally, it is aimed to maximize both precision and recall. Therefore, there exists a tradeoff between precision and recall based on the chosen confidence threshold value. A precision-recall curve plots the value of precision against recall for different confidence threshold values. To be able to find the best trade-off researchers are using the Average Precision [49]. AP (Average Precision) is a metric that combines both precision and recall into one new metric that can be used to evaluate object detection models. The name "Average Precision" means an average of the precisions across all the possible recall levels. It uses both precision and recall and creates a value of the relationship between them which is illustrated with a curve downward towards the right as in Figure 2.16 below [47].

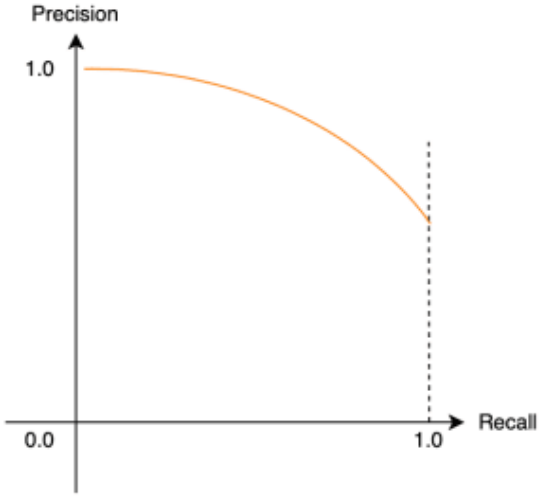


Figure 2.16: Precision-Recall curve [47]

The AP value in this graph is the Area Under the Curve or AUC. The range of AUC is also from 0 to 1. If both recall and precision is 1 then the AUC will be the area of the rectangle created and will also equal to 1.

$$AveragePrecision = \int_{r=0}^{1} p(r)\,dr$$

In real-life problems the PR-curve is not as smooth as in Figure 2.16 and there is no access to infinite data points of precision-recall combinations, so in order for the AUC to be calculated discrete intervals are used. In other words, the average of precisions from recall intervals is calculated, which is the reason behind the name Average Precision. In most cases, multiple recall values exist per interval, necessitating the interpolation of corresponding precision values. This issue is addressed by selecting the maximum precision point with a recall value $\widetilde{r} > r$, thereby determining the precision value for a given recall value r [48].

This is particularly useful as it reduces the variations in the precision x recall curve. This interpolated precision is referred to as $p_{interp}(r)$.

$$AP = \frac{1}{n} \sum_{r \in \{0,\ldots,n\}} p_{interp}(r)$$

where

$$p_{interp}(r) = \max_{\widetilde{r}>r} p(\widetilde{r})$$

and n is the number of intervals or recall points where we choose to calculate the precision.

Figure 2.17 illustrates this interpolation that was discussed before.

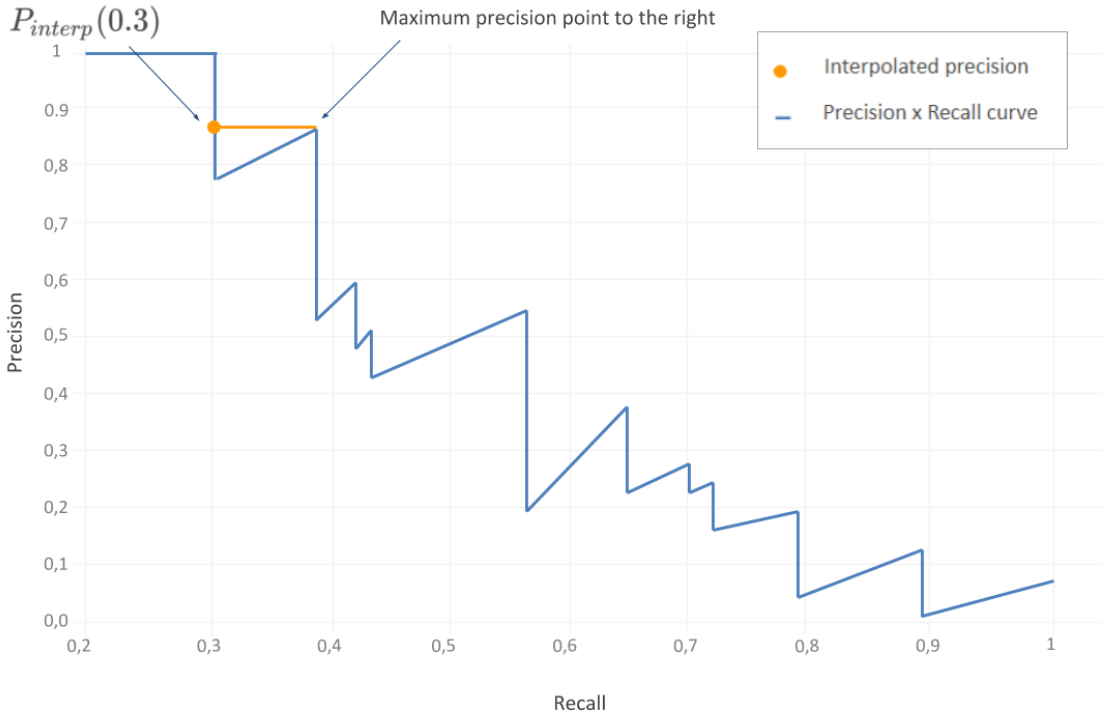


Figure 2.17: Interpolate Precision-Recall curve [48]

2.5.1.4 Mean Average Precision

Now that the AP value is analyzed and explained it's much easier to understand what the final metric stands for. So far we analyze the precision, recall, and Average Precision having in mind that only one class exists on the given problem. Usually in object detection, more than one class of objects exists and the need to expand these metrics to a multiclass problem is needed. Mean Average Precision or mAP is the average value of AP for all the given classes. In essence, if the dataset contains N class categories, the mAP averages AP over the N classes [48].

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

This is the most common metric used among object detection problems to evaluate and compare models and their results [47].

2.5.1.5 Non Maximum suppression algorithm and confidence

Before the final results are reached to the end user an algorithm used will be presented. The NMS algorithm, also known as Non-Maximum Suppression, is an important component in object detection systems. Its purpose is to eliminate duplicate or redundant object detections in order to generate a more accurate and concise set of bounding boxes around objects of interest [50]. Why NMS is used during the training process is out of the scope of this section so it will not be analyzed further. Based on the application times NMS is used after the model's predictions. The reason behind that is that in some cases the models tend to overpredict and have multiple boxes for the same object, usually most of the overpredicted boxes have also low confidence as seen in Figure 2.18a.

(a) Before NMS is applied (b) After NMS is applied

Figure 2.18: Example of the use of NMS algorithm [51]

Confidence is the probability the model thinks that the object is present in a box and it ranges from 0-1. After the NMS is applied the box with the higher confidence is kept. The IoU for two boxes to be considered identical is set by the user and it can be different from the one used during evaluation. There are some specific use cases where the use of NMS is not considered a good practice and that's when the problem consists of cluttered images with highly overlapping objects/boxes. For example for a face object detection algorithm in a crowded place, the use of NMS would not add any value since most of the objects will be overlapped [52].

In fracture detection in X-rays, NMS could add value to the results and boost the evaluation of the models since most pictures contain a limited amount of fracture and the overlap is not a common phenomenon. In most cases, the fractures are not overlapping and if they do setting a high IoU will alleviate this issue. Moreover, it also adds some value even in the case that two fractures are overlapped and the NMS algorithm eliminates one of the two boxes. This is happening because the end-user (doctor) only needs one area of interest to focus on and usually many different boxes presented in the same spot will only confuse the doctor's mind process. A better approach to this than only using an NMS algorithm would be to merge the two boxes in the previous case and not eliminate one of them, but since this is part of the final UI/UX of the application it didn't implement for this thesis.

2.5.2 Evaluation pipeline

For a better understanding for the reader, this chapter will aim to include all the algorithms that are normally used during the evaluation in a single pipeline shown in Figure 2.19.

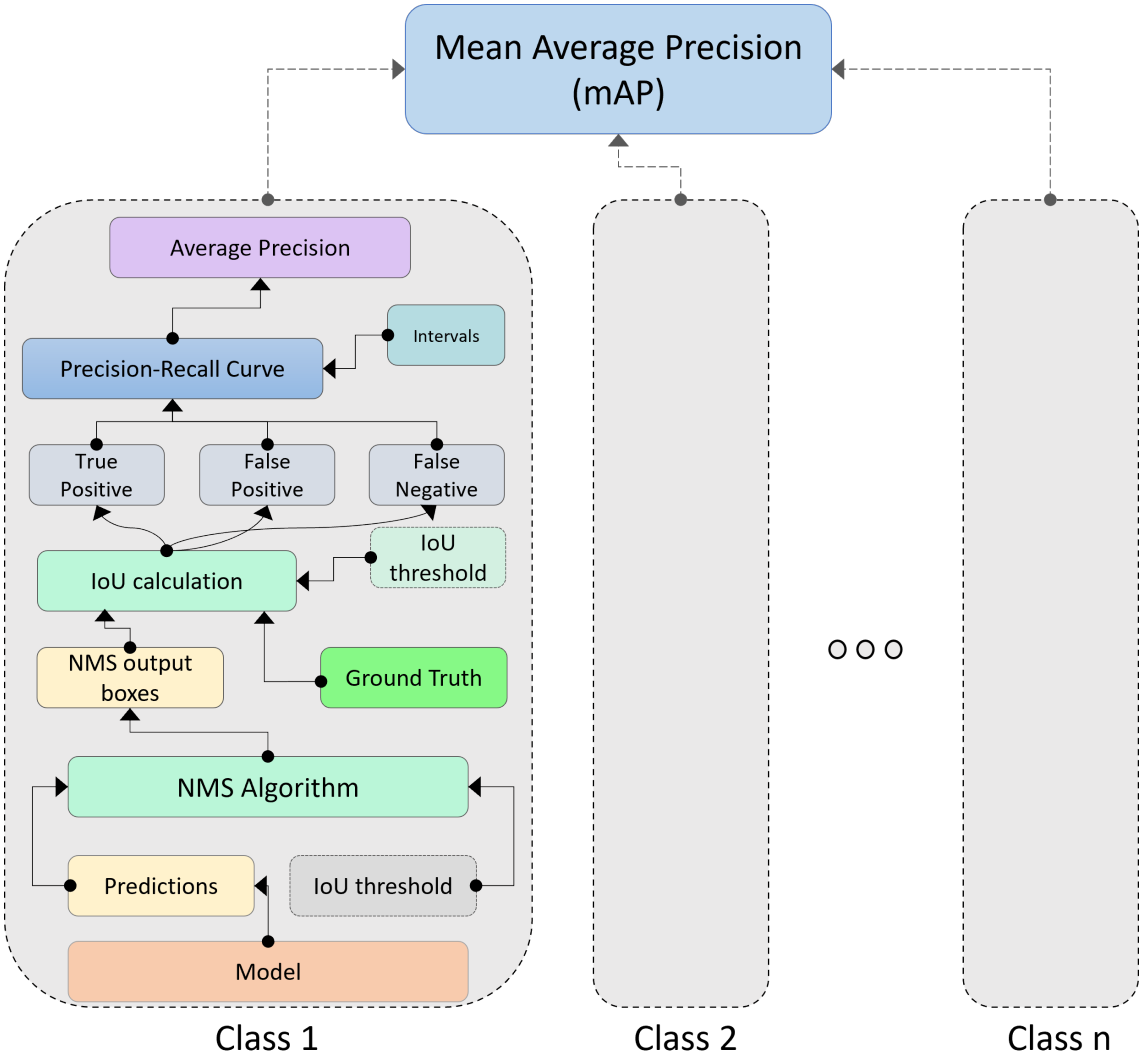


Figure 2.19: Evaluation pipeline

The evaluation process starts upon the completion of model training. The predictions generated by the model on the test set undergo the NMS algorithm. The user specifies an IoU threshold for the NMS, which varies depending on the specific application. The algorithm eliminates all overlapping boxes that surpass the provided IoU threshold. Subsequently, an IoU calculation is performed on the remaining predicted boxes and the ground truth boxes. The IoU threshold used for this calculation may differ from the one employed in the NMS. Each model prediction is then categorized as True Positive, False Positive, or False Negative. Based on these outcomes, the Precision and Recall curve is computed at specified intervals. The Average Precision is determined by calculating the area under these intervals of the PR curve. This process is carried out individually for each class among the provided classes. The mAP value, which represents the mean value of all classes, can then be calculated and typically used to evaluate the model or compare it with other models.

2.5.3 Evaluation metrics used on the experiments

In the final section of this chapter, the metrics values and the pipeline used in the experiments will be analyzed and explained. The process explained before is used in the experiments of this thesis for all the models with some minor modifications. The following Figure 2.20 presents the specific flow followed during experiments. The same pipeline and values were used for testing and validating all the models in order to have a fair comparison between them.

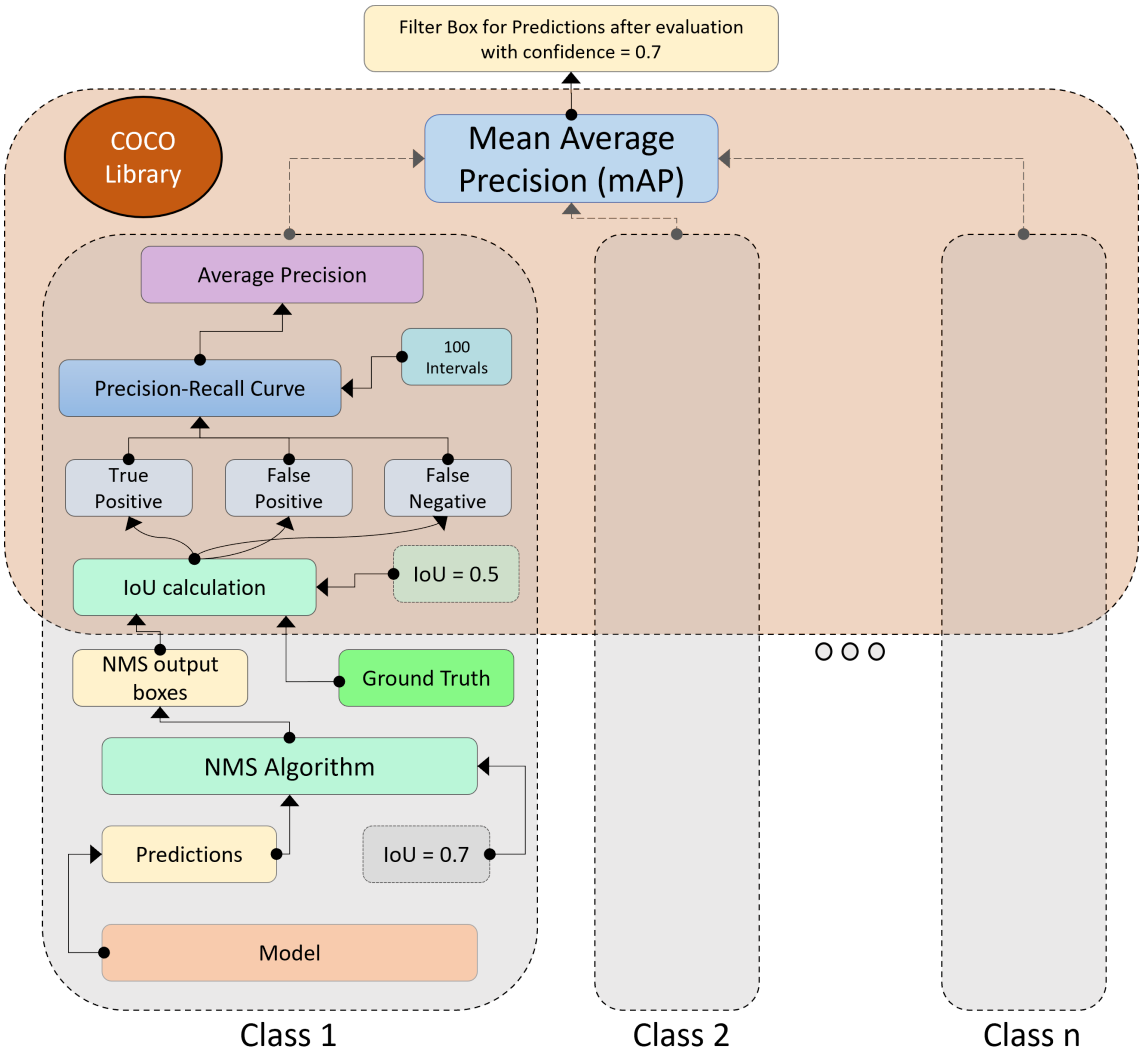


Figure 2.20: Evaluation pipeline for experiments

The main difference from the one presented in the previous chapter 2.19 is the additional filter of predictions based on the confidence of the model. This filter applies after the evaluation takes place for better visual results. After some initial experiments, a pattern was noticed when the models were validated. Most False positive predictions were made by the model with very low confidence scores usually under 0.5. For this reason, an extra step was added and after the model evaluation, all predicted objects with a confidence lower than

$$confidence = 0.7$$

are disregarded when visualization is needed. This strategy creates a better UI experi-

ence for the user. Tuning this threshold the models Precision and Recall are changed and can serve different needs of the end user. For example, usually high values in the confidence score lead to fewer False positive predictions with a tradeoff of fewer True Positives also. For the purpose of this thesis, the presented confidence had better visual results and was chosen for the Result section later. For the non-Maximum suppression algorithm, an IoU of

$$NMS\ IoU = 0.7$$

is used to filter all the overlapped boxes with an IoU higher than this threshold.

The rest of the evaluation process was implemented using the COCOeval [53] library provided by Common Objects in Context (COCO) [42]. Coco is one of the most common datasets used for object detection and around that researchers have made a single function that can be used and provide many different evaluation metrics common in object detection on any dataset as long as it is in a coco format. This dataset and its format is going to be analyzed more in the Data Description chapter. COCO calls mAP simply "AP", and they believe people should be able to tell whether it is for multiple classes or a single class from the context. The Average Precision is calculated on 100 different intervals ranging from 0.01-1 with a step of 0.01. But this is not the only metric provided by COCO. In total 12 different metrics are provided to the researchers to evaluate their models based on the application needs. Three AP values for IoU thresholds of 0.5, 0.7, and the primary metric challenge as it called calculated as follows AP@[.50:.05:.95] are provided. This last AP value takes an average of the mAP results with the IoU Thresholds from 0.50 to 0.95 increased by 0.05. Except for the typical AP value COCO also provides AP values calculated based on the area of the bounding boxes and named AP Across Scales. These metrics can help the researchers identify how their model is performing in finding small medium or large objects, depending on the problem this can be extremely beneficial. The researcher has also access to all the metrics needed to calculate the AP values such as the Recall and Precision in each interval, as well as a per-class evaluation for all the metrics.

In the process of detecting fractures on X-rays, certain metrics were considered in the final evaluation of the models, although not all of them were taken into account. It is important for the reader to bear in mind that these applications are intended to serve as supportive tools for doctors and radiologists during patient examinations. Ultimately, the doctor retains responsibility for making the final decision, and these systems aid in facilitating quicker and more accurate decision-making.

Consequently, the validity of the model's outcome lies in its ability to guide the doctor to the correct area on an X-ray. Accordingly, measures calculated with a high intersection over union (IoU) value may not be applicable in this context. Therefore, an IoU threshold of 0.5 has been selected as the primary metric for the models. Furthermore, metrics such as Recall and Precision hold significant value in the evaluation, as they assess the model's performance in terms of False Positives and False Negatives. Especially in False Negatives extra caution is needed. False negatives can result in delayed or missed diagnoses, leading to potential harm to the patient, progression of the disease, or missed opportunities for early intervention. In some cases, missing a true positive can have severe consequences, such as delayed treatment for a life-threatening condition.

3 Data Description

In the following chapter, the dataset used for the thesis experiments will be presented and analyzed. Since, the focus is an object detection problem only one dataset is used where a combination of X-ray images and annotation boxes is provided for the training and the evaluation of the models.

3.1 Pediatric wrist trauma dataset

The dataset used for this thesis is called GRAZPEDWRI-DX dataset and its an open-access pediatric wrist trauma X-ray dataset published in May 2020 [22]. The acronym is composed of the terms “Graz”, “Pediatric”, “Wrist”, and “Digital X-ray”. The dataset includes 6,091 patients’ wrist radiographs from pediatric trauma treated between 2008 and 2018 at the University Hospital Graz’s Department for Pediatric Surgery. There are 10,643 studies accessible in total (20,327 photos), most of which deal with lateral and posteroanterior projections. The collection includes 67,771 identified labeled objects and 74,459 image tags for annotation.

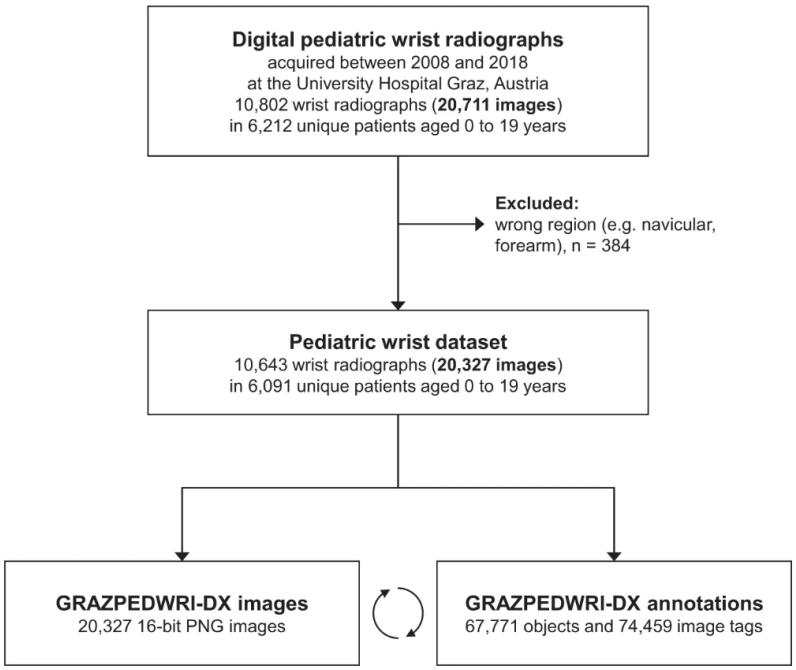


Figure 3.1: Dataset creation flowchart [22]

Fig 3.1 presents the dataset flow creation. Digital wrist radiographs were screened for eligibility in a pool of 20,711 images between 2008 and 2018 from the local ethics committee of the Medical University of Graz. Out of this evaluation 384 images were excluded due to incorrect field of view that was not focused on the wrist. The rest of the images consist of the presented dataset. This was a continuous flow through this 10-year period where images and annotations were collected and validated.

The authors employed board-certified pediatric radiologists (S.T., E.N., and E.S.) with experience in musculoskeletal radiology ranging from 6 to 29 years to validate all image annotations in order to guarantee the accuracy of the data at the annotations. This task was performed on the Supervisely (Deep Systems LLC, Moscow, Russia) [54] an artificial intelligence online platform. The web-based image database was hosted on a dedicated server, enabling collaborative tagging. Along with the aforementioned radiologists who validated the dataset, local radiologists, visiting coworkers, and medical students contributed to the dataset's advancement with various shares of labeling times. The entire annotation process took place from March 2018 to February 2022. The annotators manually placed objects with dedicated tools.

In circumstances where a fracture is known to be present but not clearly visible in the appropriate projection, Images may contain classifications without a labeled fracture object, as is explicitly stated in the paper.

3.1.1 Data formation

Each one of the images was saved in a png 16-bit grayscale format while the annotation were provided in three different formats, a PASCAL Visual Object Classes [55], a YOLOv5 format [56] and in JSON files. Information about each entry was saved in a separate CSV file as well as in each image and annotation name.

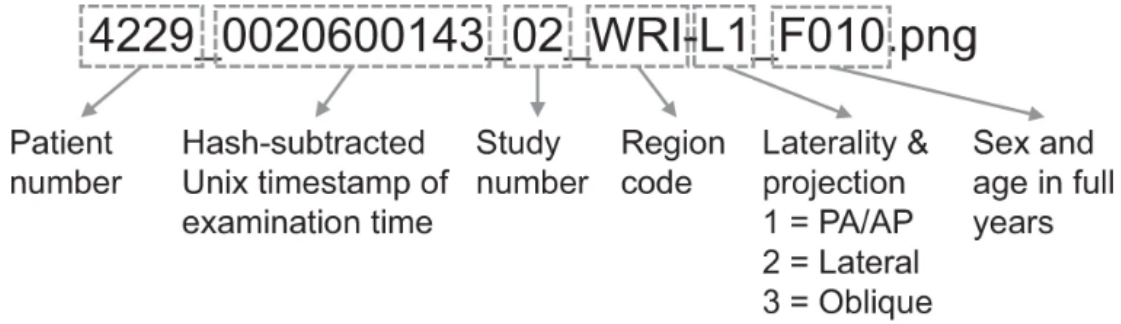


Figure 3.2: Name format of the data [22]

As seen in figure 3.2 the four-digit patient number comes first in the file names for photos and annotations, then the hashed examination time, the ascending study number, the region code (all "WRI"), the laterality and projection code, and lastly the patient's sex and age in years.

High-quality image data is given with full pixel resolution and the whole grayscale spectrum. To comply with PNG image format constraints, the writers didn't perform any post-processing other than histogram normalization, which involves expanding the typically 12-bit input spectrum to 16 bits. Because this operation is reversible, it should be noted that no information is lost.

3.1.2 Data Exploration and Distributions

This chapter will help the reader to better understand the data and the classes provided by the dataset. To initialize the exploration the following table 3.1 contains all the classes present for annotation. The percentages and the numbers of the classes as part of the total anotations are presented in the third column while the percentages of appearance in the images are presented in column number four.

Table 3.1: Numbers and percentages of annotated objects.

Object	Type	Object count	Image presence
Axis	Line	n = 20,327 (29.99%)	n = 20,327 (100.00%)
Bone anomaly	Box	n = 276 (0.41%)	n = 192 (0.94%)
Bone lesion	Polygon	n = 45 (0.07%)	n = 42 (0.21%)
Foreign body	Box	n = 8 (0.01%)	n = 8 (0.04%)
Fracture	Box	n = 18,090 (26.69%)	n = 13,550 (66.66%)
Metal	Box	n = 819 (1.21%)	n = 708 (3.48%)
Periosteal reaction	Polygon	n = 3,453 (5.10%)	n = 2,235 (11.00%)
Pronatorsign	Box	n = 567 (0.84%)	n = 566 (2.78%)
Soft-tissue	Box	n = 464 (0.68%)	n = 439 (2.16%)
Text	Box	n = 23,722 (35.00%)	n = 20,274 (99.74%)
Total		**n = 67,771 (100.00%)**	**n = 20,327 (100.00%)**

The column object shows the name of the label while the column type is the structure of the bounding box, whether it is in a box or in a polygon shape. The table indicates that the majority of the images are primarily annotated with two dominant labels: ”Fracture” and ”Text” if the ”Axis” label is not included. The Text is a label that characterizes a bounding box around a letter that exists in almost every picture and describes whether the hand belongs to the right or the left part of the body. The letter ”L” exists in pictures of left hands and the letter ”R” in pictures of right hands.

As previously stated in the introduction section, the main focus of this thesis is on the detection of fractures using a model created for this specific purpose. Consequently, during the modeling and training phases, all other classes were excluded, as predictions on these classes would have altered the original project’s focus. Additionally, the number of objects belonging to other classes in the provided dataset is quite limited, with the second most dominant class being the Periosteal reaction, with only 11% image presence. As a proof of concept and to facilitate the identification of whether or not a model has been trained correctly, the Text object was also used during the training and detection processes. This object was chosen because it exists in every picture and is easy to detect, making it a useful tool to determine if a model has been well-trained and fine-tuned to this dataset. Visual examples of all the objects will be presented in the following section.

Analyzing further the dataset and regarding the demographic aspect of it the data are gathered by a single hospital. The figure below shows the distribution of the gender of the dataset.

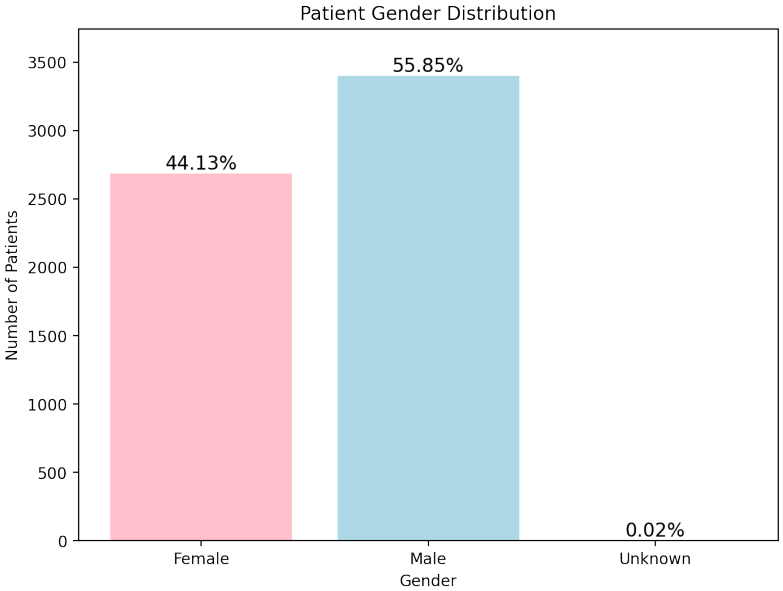


Figure 3.3: Dataset sex distribution

Since its a pediatric dataset the age of the patients range from 0.2 to 19 years old, there are two different Genders present in the data and one patient with 2 studies that didn't want to share this information. The following table 3.2 is providing a better understanding of the age ratios. Age can play a significant role in Fracture detection mainly because of the formation of the skeleton. Especially the wrist X-ray can differ a lot from a child to a young adult. This aspect of the dataset will be analyzed further in the following limitation chapter.

Table 3.2: Age description per image

Metric	Female	Male	Undefined	Total
Count	8,285	12,040	2	20,327
Mean	10.27	11.36	16.7	10.92
Std	3.52	3.59	0	3.60
Min	0.2	0.3	16.7	0.2
Max	18.7	19	16.7	19

Due to the physical variances between the sexes, there are certain differences between male and female x-rays. Females may have breast tissue that can alter the image interpretation in a chest x-ray, for instance. Additionally, the location and interpretation of x-rays might be impacted by variations in body size and shape. The bulk of the bones and organs in both men and women, however, are the same, thus the differences are typically slight and might not have an impact on the x-ray's ability to accurately diagnose a patient.

The CSV dataset that is provided with the images contains also other labels for the images that can be used in other problem and there are not that relevant for object detection. They

do on the other hand provide a good knowlage to familarize with the data and thats why the following table is provided 3.3

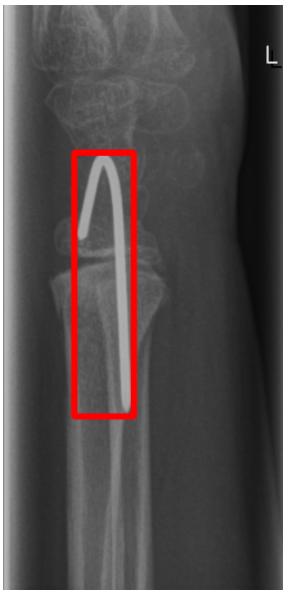
Table 3.3: Tags associated with the images

Image tag	Description	Count
Ao classification	AO classification code	n = 14,158 (69.65%)
Cast	Cast/plaster present	n = 5,776 (28.42%)
Diagnosis uncertain	Fracture labels / tags uncertain	n = 537 (2.64%)
Initial exam	Initial trauma presentation	n = 10,861 (53.43%)
Metal	Presence of metal implants	n = 708 (3.48%)
Osteopenia	Signs of osteopenia	n = 2,473 (12.17%)
Projection ap	Anteroposterior projection view	n = 10,086 (49.62%)
Projection lat	Lateral projection view	n = 10,148 (49.92%)
Projection oblique	Oblique projection view	n = 93 (0.46%)
Side left	Left side	n = 11,135 (54.78%)
Side right	Right side	n = 9,192 (45.22%)
Total		**n = 20,327 (100.00%)**

3.1.3 Visual inspection

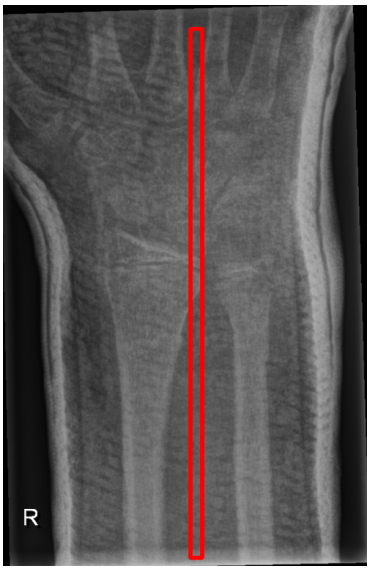
This section will include some visual examples providing an intuition to the reader about how the actual images and objects are looking. Starting in the first Figure 3.4 contains images for some of the objects present in the dataset.

(a) Two fracture objects

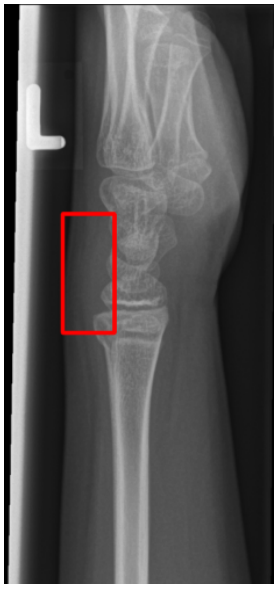


(b) Metal object

(c) The Pronator object



(d) Axis object



(e) Soft tissue object

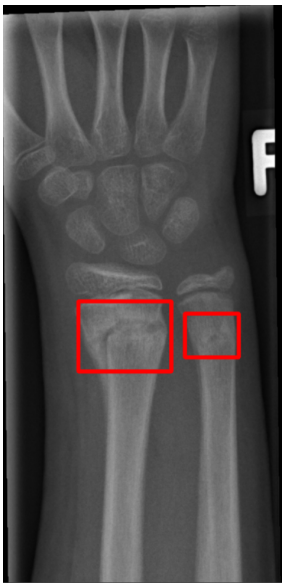
(f) Text object

Figure 3.4: Objects present in Dataset

It can be observed that some of the objects can be visible to anyone while other needs an expert to accurately tell whether or not this object is present in the image. Image 3.4a is a good example that shows that more than one object of the same class can be present in one image as well as many different objects while notably, the radiography in Figure 3.4d depicts a cast.

Focusing on the fractures, an inspection of the images can show that there is a lot that the fracture on the hand is obvious while in others a normal person will not be able to identify them. The goal of our model is to help the doctors and the radiologist to quickly identify the no obvious fractures.

(a) Image with obvious fractures



(b) Image with obvious fractures annotated

(c) Image with no obvious fractures

(d) Image with no obvious fractures annotated

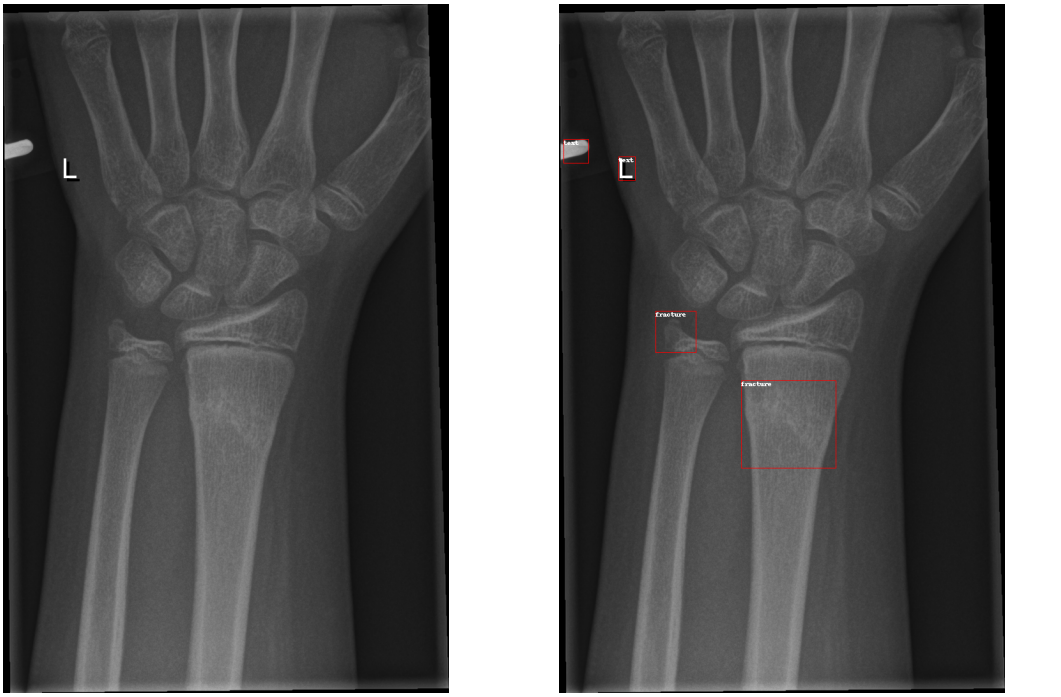
Figure 3.5: Examples of images with fractures

As previously mentioned in the introduction, radiologists are highly trained professionals who can accurately detect fractures in medical images. However, in some cases, such as the one illustrated in Figure A.1c, an AI tool may be beneficial in assisting radiologists by providing a region of interest and thereby helping to identify fractures more quickly. Moreover, an aspect that didn't analyzed before is that a tool like that can alleviate the satisfaction of search (SOS) phenomenon. The "satisfaction-of-search" phenomenon refers to a cognitive bias that can occur in radiology and medical imaging. It describes a situation where a radiologist while searching for abnormalities or specific findings in a medical image, becomes satisfied or content after identifying one abnormality, even if there may be additional abnormalities present in the same image. This can lead to the uninten-

tional overlooking of other important findings [57]. In the context of having an AI model, addressing the satisfaction-of-search phenomenon becomes relevant. By utilizing an AI model for medical imaging, there is potential to mitigate this bias. The AI model can assist radiologists in detecting and identifying multiple abnormalities or findings within an image, reducing the likelihood of missing important information and providing a comprehensive analysis.

Including the satisfaction-of-search phenomenon as a motivation behind developing an AI model in medical imaging highlights the importance of improving accuracy, reducing oversight, and ultimately enhancing patient care and diagnostic outcomes [58]. Last but not least, the use of such a tool may also improve the accuracy of fracture detection, particularly for doctors who may not have extensive experience in this area. By providing an additional level of support, AI tools have the potential to enhance the quality of medical care provided to patients.

To conclude this section an example of an image with all the annotations will be provided. This is a visual inspection of the input the models are expecting in order to be trained. The provided objects to the model are the Fractures and the Text as mentioned before. This is also the expected output of a trained model. Given the fact that the models used are trained properly this kind of output can be provided to the doctors and radiologists.



(a) Image provided for training (b) Image and annotations provided for training

Figure 3.6: Training data example

3.1.4 Problems and limitations

In this section, the limitations of the GRAZPEDWRI-DX dataset will be analyzed. Understanding the issues of the dataset can be beneficial in explaining the final results at the end while it also helps with the preprocessing of the data.

Starting this analysis the first obstacle with the data was the format. The images provided were saved in a grayscale format. An image saved with this format is an image where each pixel is represented by a single intensity value, which ranges from 0 to 255, where 0 is black and 255 is white. The image has only two dimensions: height and width. Each pixel has only one channel (or layer) representing its intensity.

Almost every deep learning model is pre-trained or construct to use RGB images. Even though this can be an obstacle to the analysis there are ways to overcome it and there are going to be analyzed in the preprocessing chapter.

The second limitation lies on the nature of the data meaning that the images are X-rays. Since X-rays are projections of a three-dimensional space, information is inevitably lost while viewing an object in two dimensions. This issue can be resolved by using MRI or CT scans, but these procedures are frequently time-consuming and expensive, hence X-rays are still a common method of fracture analysis. Besides that, doctors may have access to more information regarding each patient other than the X-ray. Oral examination or other medical examinations can help the doctor understand better if a fracture is expected to be in an X-ray or not.

Other obstacles can consider to be the number of images provided and the age limit. Analyzing in more depth the trainable images parameter the reasons behind the importance can be summed up in the following bullets [59]:

- Overfitting and Generalization: Deep learning models, with their large number of parameters, are prone to overfitting when the available training data is limited. Overfitting occurs when a model excessively matches the training data, leading to poor generalization on new, unseen data. In the case of insufficient training data, the model may memorize the training set instead of identifying underlying patterns and connections between features and the desired outcome. This lack of generalization can result in a model that fails to account for all variables and complexities in the data, leading to suboptimal performance on new data.
- Optimization: Deep learning models are optimized using a gradient descent algorithm, which requires a large amount of data to find the optimal solution. With a small amount of data, the optimization process may get stuck in a local minimum, leading to suboptimal performance.

The provided dataset is using approximately 20,000 images. One of the most well-known datasets ImageNet [40] in its most used instance ImageNet-1K contains 1,281,167 training images, 50,000 validation images, and 100,000 test images. The full original dataset is referred to as ImageNet-21K. ImageNet-21k contains 14,197,122 images divided into 21,841 classes. In object detection, the most well-known dataset is called the COCO dataset [60] and it contains 330K images (>200K labeled), 1.5 million object instances with 80 object categories, and 91 stuff categories. All in all, the available images are enough to train a deep-learning model but a bigger dataset would definitely impact the model metrics and training process.

The age limit that was mentioned before illustrates a limitation on the data generalization. The images are limited to the age of 0.2 to 19 years old. The final model might need to

predict higher age limits. Especially when the fact that the human skeleton is not the same in all ages and different characteristics exist is taken into account, it's easier to understand the importance of the age limit. Moreover, another major impact that age can have on the analysis is the structure of the bones. As Figure 3.7 shows bones are formed differently in a young child compared to an old patient.

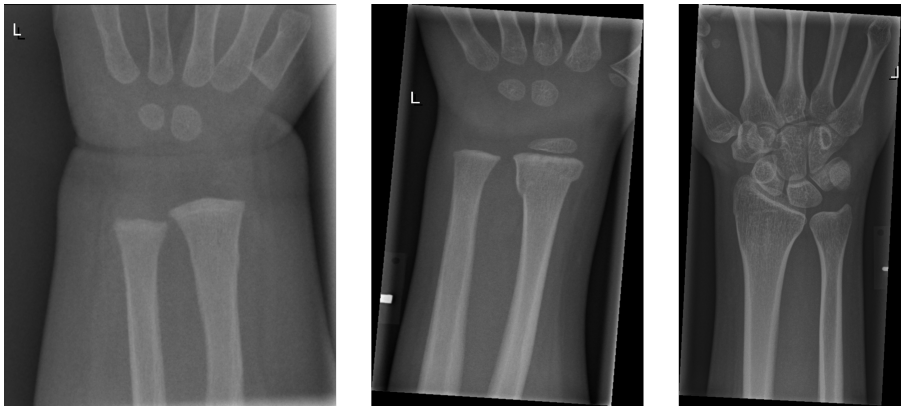
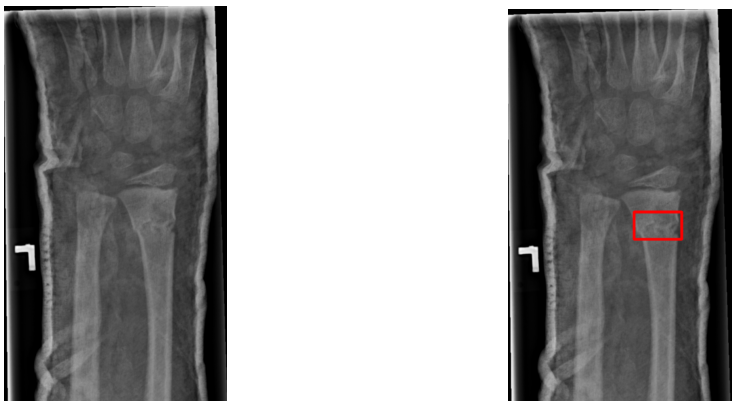


Figure 3.7: Examples of different age X-rays

The newborn baby on the left hasn't been able yet to develop all parts of the skeleton in his/her wrist, especially compared to the 16-year-old child. This can be challenging for the algorithm since it has to understand when a skeleton like this is normal and when this can indicate that something is abnormal or a fracture. This challenge is getting even bigger if the number of newborns is also taken into consideration since not many newborn X-rays are present in the dataset.

Images like Figure A.1c can also be tricky for the model to identify them. Since this model is trained to identify patterns or changes in the pixel color ratios. In X-ray imaging, the brightness of the image and the object's density are inversely correlated. Because of its high density, bone appears white in X-rays, while soft tissue like skin, body fluids, and cartilage are all black. This is one of the reasons that this model needs high-quality images to be more accurate. The following Figure 3.8 shows an example of an X-ray with a casket.



(a) Image with cast (b) Image with cast annotated

Figure 3.8: Examples of an image with cast

3.2 Preprocessing

In the final section of this chapter, all the preprocessing steps will be analyzed. The preprocessing pipeline start from the data acquisition and ends when the data are ready to be fit in the model.

The GRAZPEDWRI-DX dataset is publicly available through this link on figshare [61]. Figshare is a digital open-access repository where researchers can save and distribute their work's results, including figures, data, pictures, and videos. In accordance with the open data principle, it is free to access and free to upload content.

3.2.1 Image preprocessing

As discussed in the previous section one of the first issues that had to be solved with the data was the format. The images were saved in a grayscale 16-bit format. Fixing this issue was easy enough with today's libraries and only three steps need to be done:

- Load each image independently using the openCV library [62].
- Use a function to change the color space of the image.
- Save the image for later use.

Converting an image from grayscale to RGB can affect the quality of the image, but it does not change the image's aspect ratio or dimensions.

When an image is in grayscale format, it has only one channel of information (luminance), whereas an RGB image has three color channels (red, green, and blue). Converting a grayscale image to RGB involves duplicating the grayscale information into all three color channels, resulting in an image with the same dimensions and aspect ratio as the original. Depending on the library used the final quality of the image can be affected in different ways. For example, if the conversion is done using a simple algorithm that maps grayscale values to equal amounts of red, green, and blue, the resulting image may appear washed out or have an unnatural color cast. That's why the `openCV` library is chosen for this task and more specifically the `cvtColor` function with the input argument of `cv2.COLORGRAY2RGB`.

3.2.2 Object detection data formats

The second and bigger task of preprocessing the dataset was to configure and set the appropriate format for the images/annotations for each algorithm. In object detection, there are different formats used to map the annotation with the images as well as in the way the bounding boxes are saved.

Three of the most well-known and used formats are the COCO format [42], the PascalVoc [55] format, and the YoLo [63] format. All these formats are based on well-known object detection data sets that allow using the same format with custom datasets. There are different advantages and disadvantages for each format but in the end, they don't affect the model outcome since it's only a way to represent the data. Choosing one of those usually depends on the application and on the specific algorithm that is going to be used.

The algorithms chosen to be used in this thesis need to be provided with data in two different formats meaning that the data need to be processed two times. The DETR model supports the COCO format analyzed before while the Faster-RCNN provided by torch-vision supports a custom data format similar to the COCO but simpler. These two formats are going to be analyzed here while information about the other available format will be provided in the A.

The dataset creators provided their annotations in Pascal, Yolo and in separate JSON files one for each image. In order to create the data for the Faster-RCNN the JSON files were used and processed to create the following structure 3.9 per image that later was saved in a pickle format.

```
{'image_path': '..\\literature\\Other\\supervisely\\wrist\\img\\0001_1297860395_01_WRI-L1_M014.png',
 'target': [{'boxes': tensor([[  1., 577.,  55., 641.]]),
   'labels': tensor([2]),
   'image_id': tensor([0]),
   'area': [3456],
   'iscrowd': tensor([0, 0, 0,  ..., 0, 0, 0])}]}
```

Figure 3.9: Faster-RCNN data format

Each pickle file contains the image path and the target field where the bounding boxes, the labels, the image id, and the area of the box are saved. This structure information is fast to use and light to save. A custom dataset was created afterward to load the pickles and feed the Faster RCNN with the picture and the target field.

Preprocessing needed to be done also for the COCO dataset because the JSON file provided by the authors was per image saved. The COCO format stores the annotations in a JSON file while the bounding boxes are defined by four values in pixels [x_min, y_min, width, height]. They are the coordinates of the top-left corner along with the width and height of the bounding box. An example of the annotation is presented in the figure below :

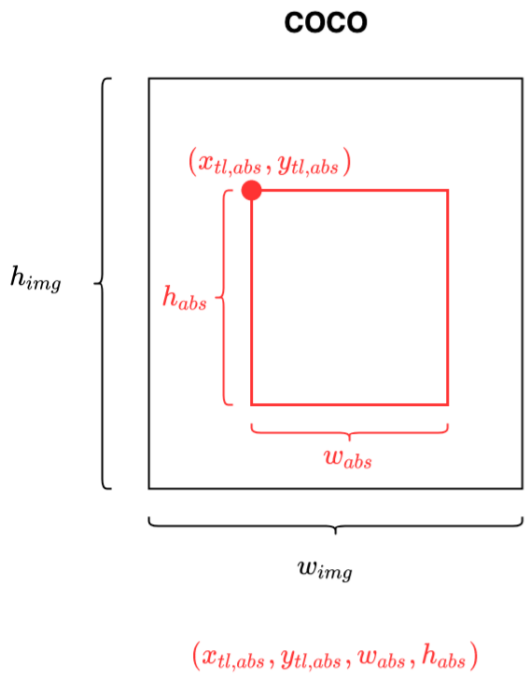


Figure 3.10: Example of COCO bounding box dimensions [64]

All the information about the images and the annotations are saved in a single JSON file easy to interpret and save.

The fundamental components of a JSON annotation file are five:

- Info: contains high-level information about the dataset.
- Licenses: provides a list of the picture licenses that are applicable to the dataset's photos.
- Categories: contains a list of categories that the bounding boxes are labeled after. Categories can belong to a super category.
- Images: contains all the image information in the dataset with the image folder and image Id to be the most important ones.
- Annotations: list of every individual object annotation from every image in the dataset. In this field, an id for the images that the annotation is belonging to is also provided along with the category it belongs to.

Both procedures were computationally intensive and time-consuming, however, the running of these functions had to be done only once and the resulting output files are efficient and user-friendly, with minimal disk space requirements.

3.2.3 Data augmentation

To effectively complete the preprocess of the dataset, applying augmentation appeared to be a reasonable step for two primary reasons that have been previously discussed. The first reason lies in the dataset size, while the second one is in reassuring our model generalization and avoiding over-fitting.

Before the pipeline is presented the reason why augmentation can alleviate these issues will be presented.

3.2.3.1 Value of augmentation

On a variety of computer vision tasks like object detection, deep neural networks as well as other type of networks have exhibited astounding performance. To prevent over-fitting, these networks, however, significantly rely on massive data. When a network learns a function with extremely high variance in order to correctly represent the training data, this is referred to as over-fitting. Sadly, many application domains lack access to big data, including a lot of applications for medical images. Data Augmentation can be a data-space solution to the issue of limited data. The term ”data augmentation” refers to a group of methods used to increase the amount and caliber of training data sets so that better Deep Learning models can be created with their aid. Augmented data is not the only method to avoid over-fitting nowadays, dropout layers, batch normalization the use of pre-trained models are just some examples of other methods that exist. The focus will be on data augmentation because except from helping with over-fitting it is also a good practice to increase the dataset size and achieve a more generalized model [65].

Diving in depth around augmentation, many different techniques can be applied including color space augmentation, kernel filters, geometric transforms, feature space augmentation, and one of the most new and evolving techniques which relies on the use of Generative Adversarial Networks (GANs) to create new ”fake” images.

As it might be clear, data augmentation is a powerful tool but it still can't solve all the problems that might exist in the data some limitations of the method can be summed up [66] :

- Quality assurance of the augmented dataset
- Verification of image augmentation techniques like GANs is challenging
- Many different strategies can be applied regarding the dataset finding one is non-trivial and can take time.
- The inherent bias of original data persists in augmented data

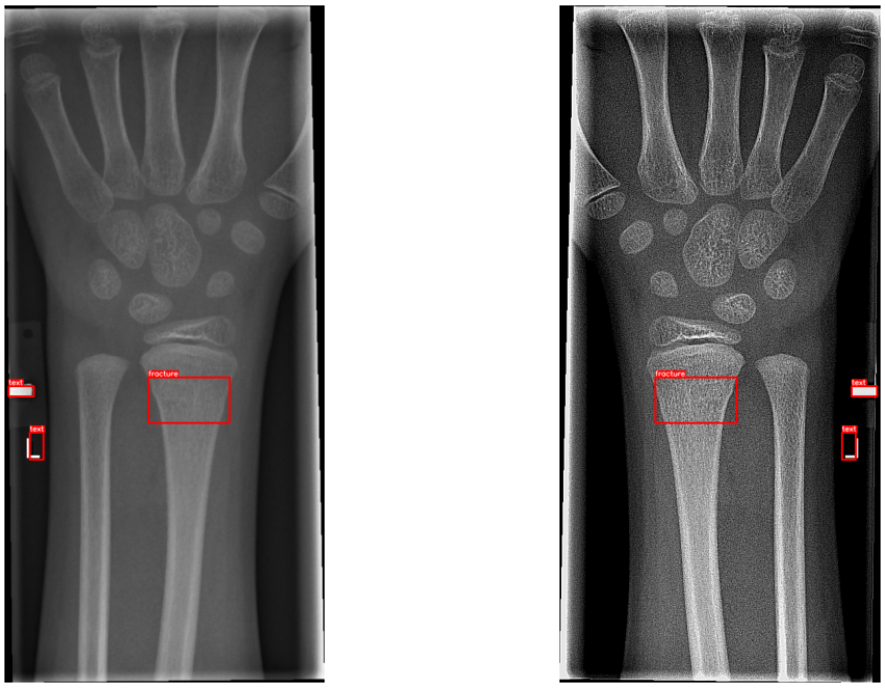
Analyzing each unique technique that can be used is out of the scope of this thesis.

3.2.3.2 Augmentation pipeline

In order to conclude this section the augmentation pipeline followed in the experiment will be analyzed and presented. Since over-fitting was not observed during the experiments the strategy followed for this thesis was simple, mostly to ensure model generalization and the extent of the training data. Following this scenario only basic image manipulation where used. To apply augmentation in an object detection dataset it's crucial to consider also the dimensions and positions of the bounding boxes and this is because changing the image can affect the box. To ensure that augmentation is working plotting the results of the faction used is essential.

The augmentation to the data is applied with the use of albumentation library. [67] Albumentations is a Python library that applies augmentation fast and reliably for different machine learning and deep learning tasks. The library was chosen because it supports Object detection tasks and is easy to interpret with torch vision [68] and other Python libraries.

After mimicking the expert's augmentation strategy of Radiobotics[69], a part of their augmentation algorithms was also applied to this thesis. The algorithms that were used from albumentation are the Horizontal Flip, Random brightness and contrast and sharpen. The HorizontalFlip flips the input horizontally around the y-axis, the BrightnessContrast algorithm Randomly changes the brightness and contrast of the input image between a fixed by-user limit and finally, Sharpen sharpens the input image and overlays the result with the original image. The motivation for using horizontal flip augmentation can be attributed to the structural similarity between the left and right hands. By flipping the left-hand image horizontally to mimic a right-hand, we can create additional training data that helps improve the model's performance on right-hand images. This augmentation technique takes advantage of the fact that the structure and features of hands are generally symmetrical, and by introducing mirrored versions of the training data, the model can learn to better handle variations and orientations specific to right hands. This helps enhance the model's ability to accurately classify and detect right hands in real-world scenarios. All these transformations were used through the data loader. An example of an image transformed is shown in the next Figure 3.11



(a) Original Image (b) Augmented Image

Figure 3.11: Original image and Image after augmentation

Albumentation offers a parameter in each algorithm that the programmer can define whether or not the transformation is going to be applied in all pictures. If the is set to false then with a chosen probability the transform will or applied to the data. In case the first option was chosen, applying the transformation in each image then going through the whole training set applying the transformation and saving the image would be necessary to do in order to train the data in both the original set and the augmented one. This is computational and space-consuming. For this reason, the second strategy was chosen. Every time an image is loaded the is a probability set to p = 0.5 for each transformation to be applied or not. This in combination with setting more epochs for the model in training means that at some point the model will be trained in both the original data and the augmented one.

This is the strategy followed during the experiments to overtake the obstacle of having a small dataset and to avoid potential over-fitting and under-generalization of the model.

4 Modeling

After reviewing the project's motivation, the architecture of the models utilized, and the data that will be processed in this thesis, all the essential components of this project have been established. This chapter will specifically focus on the modeling aspect and how these models were employed to construct an end-to-end deep learning architecture providing a valuable tool for hospitals and healthcare practitioners.

4.1 Project set up.

Before the main modeling part will be presented a deep dive into the tools used and how can an individual reproduce all the experiments will be elaborated in this chapter.

4.1.1 Environments and Development

The initial requirement for this project was to establish a dedicated Python environment. These environments facilitate the installation of various Python packages that contain functions relevant to the project. Furthermore, they handle the management of dependencies, ensuring the project's reproducibility. For the purpose of this thesis, a specialized Conda environment was specifically created [70]. Conda is a package management system and environment management system that is open source and compatible with Windows, macOS, and Linux operating systems. It efficiently installs, runs, and updates packages and their dependencies. Additionally, Conda enables the creation, storage, loading, and switching of environments on your local computer. The editor used for the project was a Visual Studio Code editor and Python 3.8.8 was the programming language since both of them can provide everything a researcher needs to develop deep learning models. [71, 72] The project structure inside the VS code was initialized using Cookiecutter. Cookiecutter is a command-line utility and project template system that allows for the generation of project scaffolding from pre-defined templates. It is designed to automate the repetitive setup process when starting a new project by providing a consistent and standardized directory structure, file templates, and configuration files. Using Cookiecutter, developers can quickly create new projects with a predefined structure and pre-populated files. The templates used by Cookiecutter are typically stored in version control repositories and can be customized to fit specific project requirements [73].

4.1.2 Version Control and Reproducibility

Effective collaboration, sharing, and maintaining version control in projects are crucial aspects, especially when it comes to code and project management. It is essential to have a system in place that enables easy sharing, tracking of code versions, and documenting changes. Furthermore, for experiments to be reproducible, it is vital to have a clear understanding of the code used for each experiment.

To address these requirements, Git and GitHub were utilized [74, 75]. Git is a distributed version control system that allows for efficient management of code changes and collaboration among team members. GitHub, on the other hand, is a web-based hosting service that provides a platform for storing Git repositories and facilitating collaboration and project management.

By leveraging Git and GitHub, this project ensures version control, Reproducibility, and a secure repository for all code-related assets. It enables seamless tracking of code changes, simplifies collaboration among team members, and provides a centralized platform for storing and sharing code. This combination of Git and GitHub supports the

project's objectives of version control, Reproducibility, and maintaining a well-organized codebase.

4.1.3 Experiment logging

In addition to the project setup, another crucial aspect is the implementation of a tool to effectively manage and monitor experiments along with their results. To ensure successful logging, two different tools are utilized. Firstly, as mentioned in the previous chapter, Git and GitHub are employed to keep track of experiment hyperparameters and code changes. While there are more specialized tools available to track hyperparameters, such as utilizing a configuration file to store them, it was deemed unnecessary for this project due to the relatively small number of parameters to tune. Each experiment was recorded as a separate push command to the main branch in Git, accompanied by a descriptive message indicating the nature of the experiment.

For experiment logging, another tool called Weights and Biases was employed [76]. Weights and Biases provide a comprehensive platform for experiment tracking, visualization, and analysis. Given that all the experiments were executed in a High-Performance Computing (HPC) cluster using a platform like this allows the user to have access anytime and anywhere of experiments and to be able to see live the results and the logs that are chosen to be saved. All the basic hyperparameters were also saved in W&B since it offers tools and functions to keep track of many different parts of the executed code.

By utilizing a combination of Git/GitHub for code and hyperparameter tracking and Weights and Biases for experiment logging, the project ensures systematic management, traceability, and effective monitoring of all experiments and their outcomes.

4.1.4 Data infrastructure

Going through the Data infrastructure, since the same dataset and version of the data were the same from the beginning of the thesis till the end all the Data storage and manipulation were done locally. Python notebooks were employed for data transformations, ensuring the data was prepared for algorithms. Once this process was complete, a free and open-source FTP client software called FileZilla [77] was utilized to transfer the local data to the HPC server. The researcher was provided with storage space on the DTU (Technical University of Denmark) server, making this transfer possible. Although this process was time-consuming, it only needed to be performed once. Additionally, FileZilla was utilized during the experiments to transfer models and run logs from the DTU server to the local environment, enabling result replication.

4.1.5 MLOps pipeline

MLOps stands for ”Machine Learning Operations.” It is a practice that combines machine learning (ML) and operations to streamline the development, deployment, and management of ML models in a production environment. MLOps aims to address the challenges and complexities associated with the lifecycle of ML models, including data preparation, model training, version control, testing, deployment, monitoring, and maintenance [78].

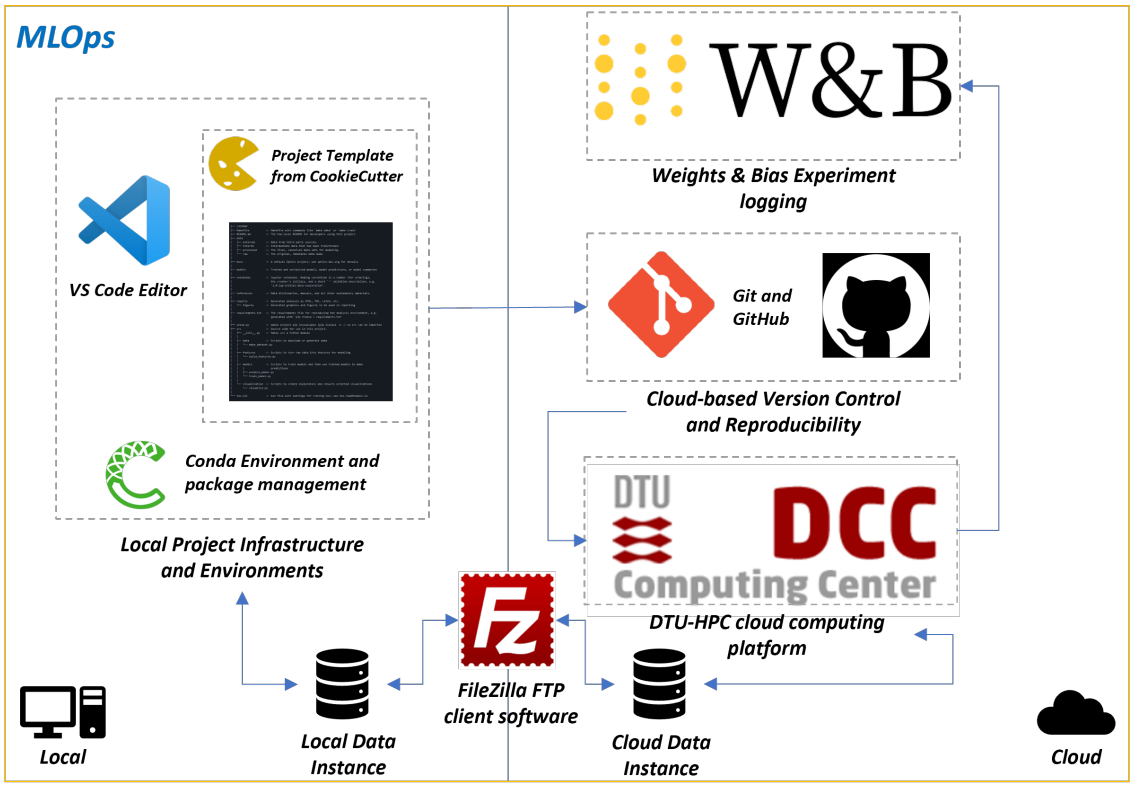


Figure 4.1: The complete MLOps pipeline

Combining all the different tools explained in the previous sections the Machine Learning operation pipeline presented in Figure 4.1 is generated providing an end-to-end environment for performing experiments and creating deep learning models while at the same time, it ensures reproducibility, versioning of environment and code as well as experiment logging.

4.2 Single-View approach set-up

This section will provide all the necessary details of the training pipelines that were the same for all the models before the specific model implementations are presented. One of the most challenging aspects of the whole project was to ensure that all the models would be comparable at the very end, to either validate or not the hypothesis that a ”context-aware” model can be competitive compared to Faster RCNN architectures.

The most important part to ensure comparability is the data. It is essential to use the same data for training testing and validation since the quality of the training and the testing data can affect the model performance the most. This was implemented by splitting the

data patient-wise. Each patient can be present in multiple studies while at the same time, each study can contain from one to four different X-rays depending on the doctor's need. To ensure consistency in model training and prevent any data leakage from the training set to the test set, a custom data-splitting strategy was employed. The data was partitioned in such a manner that if a patient's data was included in the training set, all of their corresponding studies were also included, while none of their X-ray images were present in either the test or validation set. This approach guarantees that each patient's data remains confined to a single set, maintaining the integrity and independence of the evaluation process. After reassuring this the data was split into 75% train data, 15% testing data and 10% validation. The data for the FRCNN model in the dictionary format was first created and then based on that each set was transformed to the corresponding COCO dataset.

Following the same logic the second globally set hyperparameter was the Batch size. The batch size signifies the number of samples or data points processed as a collective unit in one iteration of both forward and backward propagation during the training of a Deep learning model. Essentially, it represents a subset of data utilized to update the model's weights in each iteration or epoch. There is various research on how batch size can affect the model metrics but in general, is a parameter that can vary, and finding the best number also depends on the provided resources. Since training a detection transformer is more computationally hard compared to a CNN-based model the Batch size was set at 16 because with higher values the model often exited the training face due to lack of memory, even though the FRCNN was able to run without any problems also at 32 [79].

The final step involved selecting the appropriate models for comparison. To ensure a fair and meaningful comparison between the two different architectures, careful considerations were made. To achieve this, both models were implemented using the PyTorch framework [80] even though implementation of these models in other frameworks does exist.

4.2.1 F-RCNN

Moving to the model set-up the training process of the Faster-RCNN will be presented first. For the model, a custom dataset class was created dedicated to the task of loading the image and the annotations from the pickle file that were saved and applying the augmentation pipeline. The model was instantiated using the default weights of Faster-RCNN provided by TorchVision. The training functions the data files and all the utils functions were pushed to the HCP environment offered by the DTU University and all the experiments were processed there. Since object detection is a very computationally hard task all the experiments used the GPU infrastructure and more precisely the Tesla A100 PCIE 80GB or 40GB. All the results were saved directly to the Weights&Bias while the model itself was saved locally to the HPC and retrieved after the execution. All the FRCNN models converged quite fast in the first 20 epochs but the run was usually performed for 100 epochs in total and took almost 24 for the training to complete.

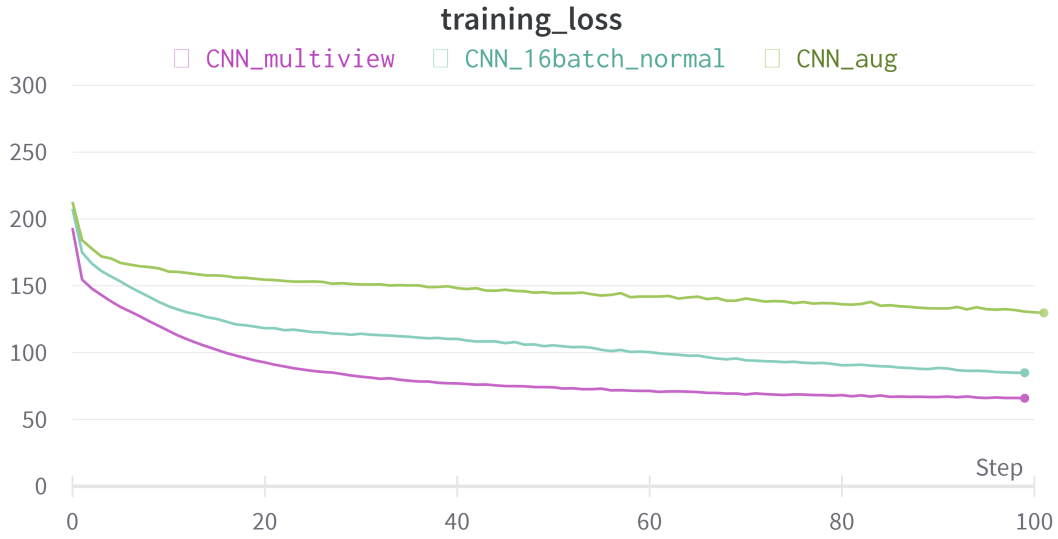


Figure 4.2: The FRCNN training loss for Multiview Augmented and not Augmented model

In Figure 4.2 the training loss of the 3 last models for the single view with and without augmentation and for the Multiview problem also illustrates the fact that the model converges fast.

After each epoch during training, a validation step was initiated to assess the model's performance. The coco evaluation function was applied to the model's predictions on the validation data, computing various metrics as discussed in the metrics chapter. If the AP50 value (a specific metric) surpassed the previous best value, the model's state was saved. This approach enabled the identification of the highest-performing model on the validation set, which would then be utilized as the final model for subsequent testing. The process of performing a validation step during training and saving the best-performing model holds significant value in model development. By evaluating the model's performance on a separate validation dataset, this technique enables effective model selection based on specified evaluation metrics. It helps prevent overfitting by identifying the point at which the model starts to specialize too much on the training data. Ultimately, this technique provides a robust approach to identify the best-performing model and confidently proceed to the testing phase with a strong candidate that demonstrates superior performance on unseen data, since if the model at the final epoch were saved the results maybe the loss would be lower but the performance on unseen data could be worst.

In each experiment, different hyperparameters were tunned trying to find the best-performing model. Even though there are tools for helping the researcher perform hyperparametaratiation tuning like Weights&Bias sweeps these were not applicable for this thesis since all the experiments needed to take place in the DTU HPC and the computation times were too high reaching most of the time the running time limits of the HPC cluster. As a result, a number of different values were tested and the best-performing model was saved at the end.

Table 4.1 displays the tested hyperparameters for the FRCNN model, with the ones that performed the best highlighted in bold. The same set of parameters was utilized to evaluate and compare the model with and without augmentation. Consequently, in the subse-

Table 4.1: Hyperparameters Tested

Parameter	Possible Values
Learning rate	{0.1, 0.01, **0.005**, 0.0005}
Optimizers	{AdamW, **SGD**}
Momentum	{0, **0.9**}
Weight decay	{0.001, **0.0005**}
Gradient Clip	{**No clipping**, 1}

quent chapter with the results, all the models were trained using the bold hyperparameters presented in the table.

4.2.2 DETR

The DETR model setup is similar to the FRCNN approach. A different custom dataset was implemented for this architecture since an extra step needs to take place for this of DETR image processor. The DetrImageProcessor class is a component used for processing images in the context of the DETR (DEtection TRansformer) model. It provides various functionalities for preparing images before feeding them into the DETR model for object detection. Despite this change in the dataset, all the other steps are similar to the FRCNN implementation and the augmentation pipeline follows this step. All the experiments run also in the HPC cluster of DTU and on the same GPUs. The difference during the training face was that the DETR models needed more than 300 epochs to converge meaning 2-3 days of running on the GPU cluster. That created different issues in the training process mainly because of the memory usage and the time limit resulting in timed-out runs by the HCP. To alleviate this problem smaller runs were performed saving the model at the end and reloading it again to continue the training.

The training curves below in Figure 5.14 are illustrating this approach. This is a single model run divided into 5 smaller runs of 90 epochs each.

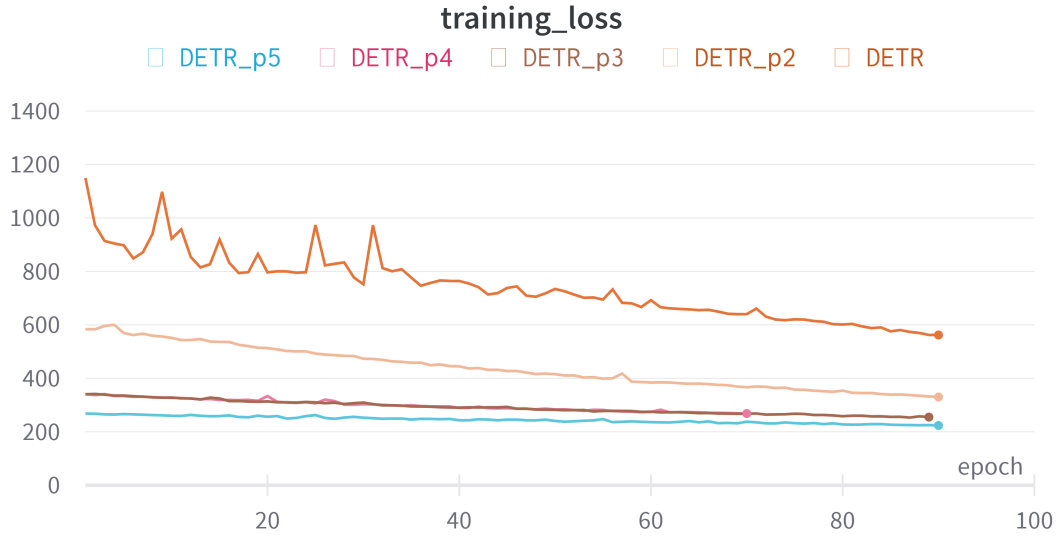


Figure 4.3: The DETR training loss divided into 5 different runs where each end value of DETR_p$_X$ is equal to the start value of DETR_p$_{X+1}$

The DETR loss started way higher compared to the one the FRCNN had and this can be explained by the way the two models calculate the loss. As stated on the model's architectures the FRCNN loss is a weighted sum of the RPN and the total network classification and regression loss while on the other hand, the DETR model calculates the loss as a classification, a regression loss, and a Set-based Loss. DETR introduces a set-based loss, such as the Hungarian loss or bipartite matching loss, to establish correspondences between predicted and ground truth objects. This loss encourages one-to-one matching of objects and helps handle cases of variable numbers of objects in an image. So the two loss functions are not comparable.

The same validation strategy was followed during the DETR training process meaning that the model that was forwarded for testing was the one performing the best on the validation set.

The DETR model hyperparameters are presented in the Table 4.2 below:

Table 4.2: Hyperparameters Tested

Parameter	Possible Values
Learning rate	{ 0.001, **0.0001**, 0.0005}
Optimizers	{**AdamW**, SGD}
Momentum	{**0**}
Weight decay	{**0.0001**, 0.0005}
Gradient Clip	{No clipping, **1**}

The model was trained with a stable learning rate of 1e-5 for all the experiments for the backbone network as the authors of the DETR model did in their implementation since the results were quite unstable if this wasn't set during training.

4.2.3 Conclusion on Hyperparameter tuning

Concluding from what was presented in the previous two sections, hyperparameters are one of the biggest challenges in deep learning since it is not as easy to tune as in other Machine learning tasks but they do affect a lot the final outcome of the models. Both models were affected a lot by the value of the Learning rate and by the choice in the optimizer. The parameters that ended up used for the models are really close in both cases to the ones they had in the original papers' implementation. For the faster RCNN, the parameters are exactly the same with the only difference being in the momentum whose value was not clear in the original paper. The model performed poorly in high values of learning rates and with the AdamW as an optimizer. In contrast, the DETR model performed worse in lower values of the Learning rates while it's training, in general, was more unstable compared to the FRCNN since many times the training loss had spikes, and that's also the reason that the Gradient clip was implemented which is not present in the original implementation. In this model, SGD didn't have the expected results and the AdamW optimizer met the expectations having a significantly better performance compared to the other. All in all, there is more space in optimizing the hyperparameters if needed since there are a lot more values that can be tuned or combinations to be tested. For the metrics presented here, every optimizer was tested with all the Learning rates present, and after that for the best performing models the other metrics were tuned.

4.3 Multi-view approach

In order to compare the models trained using the single-view approach, a similar setup was followed for the Multi-view approach. The key difference lies in the data used, as now one or more images are combined, depending on the specific study. However, the patients used for training, testing, and validation remain the same as those in the single-view approach. This ensures that all models, regardless of the approach, are trained on the exact same set of images and tested on the same set as well.

Regarding the rest of the pipeline, there were some adjustments made to accommodate the new data in the Multi-view approach. Different dataset classes were utilized to load the modified data, but aside from that, the overall process remained consistent. However, due to time constraints, an augmentation pipeline was not implemented for the Multi-view models. Despite this, the model that achieved the highest validation score was saved and subsequently used for the testing phase.

It's worth noting that the Multi-view models took longer to converge compared to the single-view models, likely due to the increased complexity introduced by combining multiple images. Nonetheless, by following a standardized approach and utilizing the same set of patients, a fair and meaningful comparison can be made between the models in both the single-view and Multi-view approaches.

5 Results and Discussion

The final chapter of this thesis is devoted to the results and discussion of the experiment. It is assumed that the reader has gone through all previous chapters. This chapter structure has divided into three different main sections. The results of each approach are going to be discussed independently and in the final section, a comparison between all models and approaches will be performed.

5.1 Single view approach results

As discussed in the previous chapters the models during the single view approach consider every image as an independent entity. The predictions are per image and the relation between patients and studies is not taken into consideration. The results will be presented both collectively and individually for each class. This approach is adopted because the fracture class carries the utmost importance in this context.

Additionally, the 'Text' class has been included as a baseline class for object detection. Its purpose is to validate the training of the models and assess their proficiency. Since identifying the 'Text' class should be relatively easy for these models, it is expected that its AP@50 value, along with all other metrics, will be considerably high. Consequently, the metrics will be presented separately for each class to highlight the model's ability to predict fractures accurately. Although the 'Text' class may increase the overall AP@50 value and make the model appear better, it does not necessarily reflect its capability to predict fractures effectively.

In summary, the results will include an AP@50 value specifically for fractures, which will serve as the determining factor in identifying the best model.

5.1.1 Faster-RCNN results

This section covers the results of the FRCNN model. The best-performing models as discussed in the previous section are going to be presented for the models with and without augmentation to also show the importance of this aspect of the project.

Model	Label	AP@50	mAP@50
F-RCNN augmentation	Text	0.9883	-
	Fracture	**0.8997**	-
	Total	-	**0.944**
F-RCNN	Text	0.9883	-
	Fracture	0.8815	-
	Total	-	0.935

Table 5.1: F-RCNN Results

Table 5.1 presents AP@50 scores for each class, as well as the overall mAP@50 value for the model. Upon analyzing the table, it is evident that the 'Text' class is easily detected by the model, confirming the initial assumption. Consequently, it is advisable to exclude the 'Text' class from the final evaluation of the model. This exclusion is necessary to ensure a more accurate assessment of the model's performance, as including the 'Text' class inflates the total mAP@50 value, potentially misleading the perception of the model's true

capabilities. This is the main reason that the discussion from now on will focus on the fracture scores.

There it can be observed from the results that both models with and without augmentation reach quite similar and impressive results with an AP@50 value of 89.9% and 88.1%. Even though the difference is small the value of augmentation is depicted in the results with the augmented model achieving almost a 2% rise to the AP value of the model in fracture detection. The small difference between them raises the possibility that if the average value from multiple runs were calculated, the results would likely be indistinguishable. However, due to time constraints, this minor detail was not explored further. Despite the limited analysis, it is evident that both models exhibit similar levels of accuracy, reinforcing the notion that they are comparable in performance.

To better present the final results and to be able to compare them more accurately the Precision-Recall curves will be presented. The area under these curves represents the AP value.

Similar to the results in Table 5.1 the curves for both classes will be presented but the emphasis of the analysis will focus on the Fracture curve. The curve for the non-augmented model is presented in Figure 5.1 and the curve for the augmented model in Figure 5.2. In addition to the IoU threshold of 0.5, it has also been included the curves for IoU thresholds of 0.7 and 0.9. This deliberate choice aims to highlight the significant impact that this value has on the final results. By including these additional curves, it strengthens the argument made earlier that the commonly used AP50:95 mean value in object detection is not applicable or meaningful in the context of this particular application.

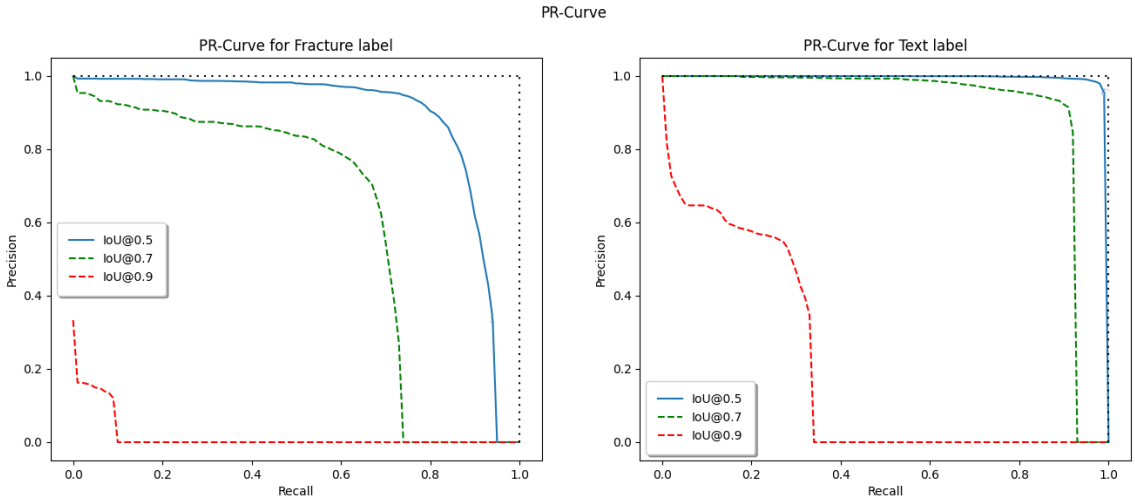


Figure 5.1: Precision-Recall curve of FRCNN model without Augmentation

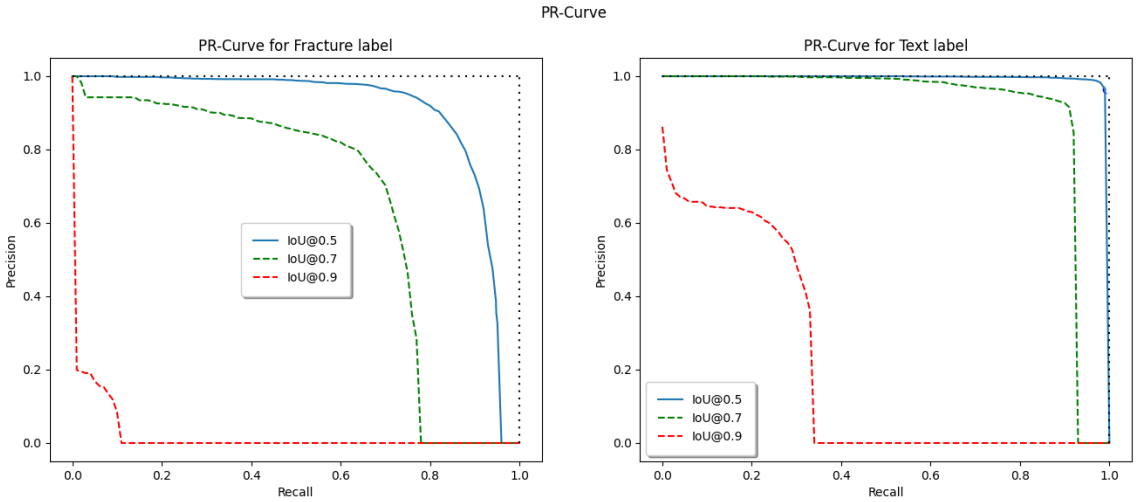


Figure 5.2: Precision-Recall curve of FRCNN model with Augmentation

The blue line in the graph represents the AP values for the IoU threshold of 0.5. While both models have similar AP values, it is evident that the augmented model's curve covers more area under the curve compared to the non-augmented model. This distinction, although minor, indicates the improved performance achieved through augmentation.

By traversing along the curve and adjusting the confidence threshold, users can select models with varying precision and recall. This flexibility becomes particularly advantageous when deploying the model, as a single parameter adjustment can cater to different user requirements. For instance, in certain scenarios, allowing for some false positives may be acceptable if it results in higher true positive rates. On the other hand, different combinations can serve alternative purposes. High precision is desirable in scenarios where false positives have significant consequences like in the case of fracture detection where this would lead to unnecessary treatment and expenses. Lower recall implies a higher occurrence of false negatives, which can have severe consequences, particularly in medical scenarios where patients in need of treatment may go undetected. This

prompts the question of whether the trade-off is justified, given the substantial improvement in precision compared to the decrease in recall.

Additionally, an intriguing observation from the Fracture curves is that, for various thresholds and low sensitivity (recall or true positive rate), the model demonstrates a Precision of 1 or close to it. This suggests the existence of models that accurately predict all existing fractures, which may not always be the case at lower recall values, especially for that big of a range for different recalls.

Furthermore, the curves demonstrate that the model predicts text accurately in nearly all instances, as evidenced by the consistently high precision and recall values associated with the ”Text” category.

Overall, someone could argue that the augmentation did not significantly enhance the model in terms of the average precision (AP) value but what needs to be considered is that the values without the augmentation were already high leaving less space for improvement. It is undeniable though that the model created with augmentation contributed to a more robust model that is better equipped to handle variations and exhibit improved generalization capabilities.

5.1.2 Detection Transformer results

Following the same structures as the previous chapter, the results of the DETR model will be presented and discussed here. The best-performing models as dis- cussed in the previous section are going to be presented for the models with and without augmentation :

Model	Label	AP@50	mAP@50
DETR augmentation	Text	9653	-
	Fracture	**0.7803**	-
	Total	-	**0.904**
DETR	Text	0.9505	-
	Fracture	0.7237	-
	Total	-	0.869

Table 5.2: F-RCNN Results

The results clearly demonstrate the positive impact of augmentation on all classes and metrics. The models enhanced with augmentation exhibit a remarkable mAP@50 of 90.4%, surpassing the non-augmented model’s mAP value of 86.9%. As stated before the results in the ’Text’ label are good enough to affect the final mAP value and thus there are going to be excluded from the final decision of the models. However, the most notable improvement lies in the ’Fracture’ label, as evident from Table 5.2. The augmented model showcases a noteworthy increase of 6% in the AP value. This considerable enhancement underscores the effectiveness of augmentation in capturing the intricacies of the ’Fracture’ label and enhancing model performance.

Transformer models are widely recognized for their inherent complexity, resource-intensive training process, and reliance on substantial amounts of data to achieve exceptional performance. However, in this particular scenario, it has been conclusively demonstrated that augmentation can serve as an effective strategy to expand the size of the training dataset by creating additional samples with more diverse examples to learn from, leading

to enhanced model generalization and performance. The results leave no room for doubt as the augmented model clearly outperforms its non-augmented counterpart.

The same Presicion-Recall curves will be presented also for the DETR models aiming to enrich the results for the reader.

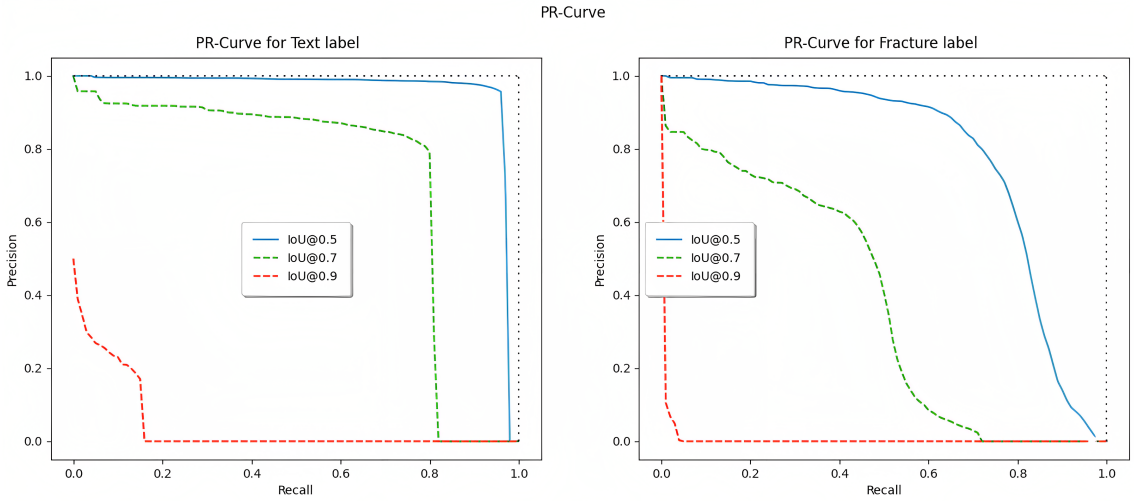


Figure 5.3: Precision-Recall curve of DETR model with Augmentation

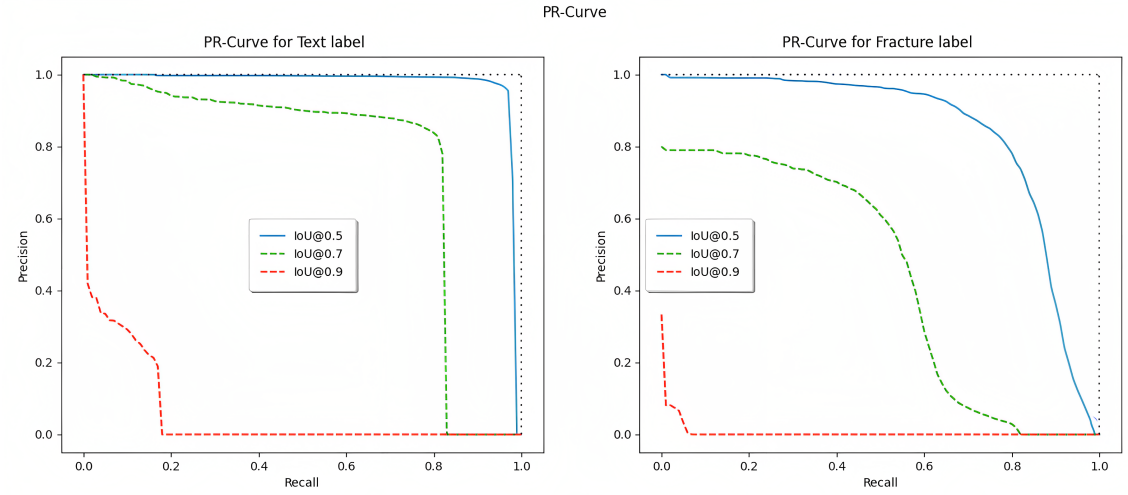


Figure 5.4: Precision-Recall curve of DETR model with Augmentation

In the presented Figures 5.3,5.4 with the curves for the DETR models it is more obvious how augmentation has affected the model than it was with the FRCNN since now the difference in the AP value is bigger. A significant contrast is evident in the augmented PR-Curve concerning the model's response to lower sensitivity (recall or true positive rate). The augmentation technique helps the model achieve a precision of 1 at lower recall values for various thresholds, thereby improving the performance compared to the non-augmented approach. In the non-augmented case, the precision dropped almost instantly for recalls different from 0. This finding further reinforces the observation made in the previous paragraph regarding the performance of the FRCCN model at lower recall values. Comparatively, the FRCNN model demonstrates even better performance in

terms of precision, proving that achieving a precision of 1 is not common across a wide range of recalls.

5.1.3 Comparison

This chapter aims to compare the performance of the FRCNN and DETR models and provide a comprehensive analysis of the results obtained. Through this discussion, valuable insights and key findings will be uncovered regarding the effectiveness of the approach employed. The collective results are presented in Table 5.3

Model	AP@50
DETR augmentation	0.7803
F-RCNN augmentation	**0.8997**

Table 5.3: Model Performance Comparison on 'Fracture' detection

As stated before the model comparison will be performed only on the fracture label since this is the important one. Both models performed equally well for the 'Text' and since the mAP value is affected by this performance the presented metric in Table 5.3 Average Precision is for the 'Fracture' class only. The superiority of the Faster RCNN model over the Detection Transformer becomes readily apparent upon closer examination of the provided metric. While it is true that, in comparison to other literature findings outlined in prior chapters, one could argue that both models compete well when compared to other cutting-edge models, it is important to note that there exists a significant disparity of nearly 11% in the AP value between them. Although this discrepancy may seem trivial in other projects and applications, within the domain of medical applications, the significance it carries is profound. Each incremental enhancement in performance directly translates into improved patient care and overall advancements in healthcare. As such, every single point holds profound importance in this context.

5.1.4 Visual Results and Discussion

Given the project's primary focus on fracture detection, it is crucial to present visual results to the reader. These visual outputs serve multiple purposes, including facilitating a better understanding of the models' outputs and enhancing the discussion surrounding the topic and models at hand. By incorporating visual representations, the project gains clarity and richness, ultimately improving the overall comprehension and analysis of the fracture detection models.

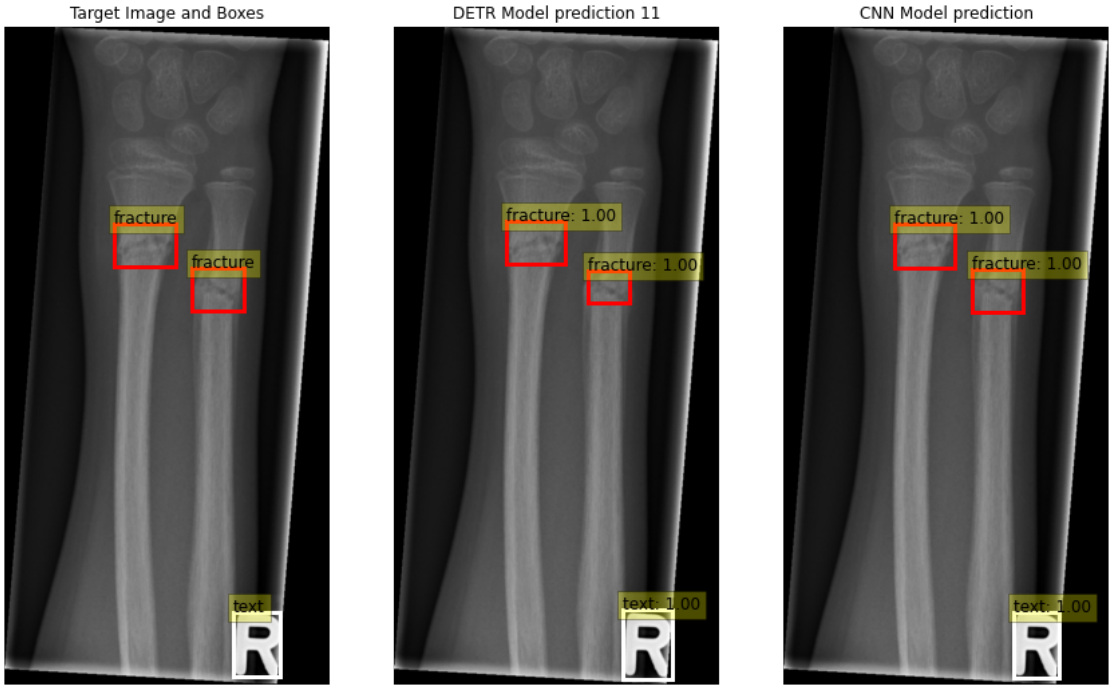


Figure 5.5: Example of detection on obvious Fractures

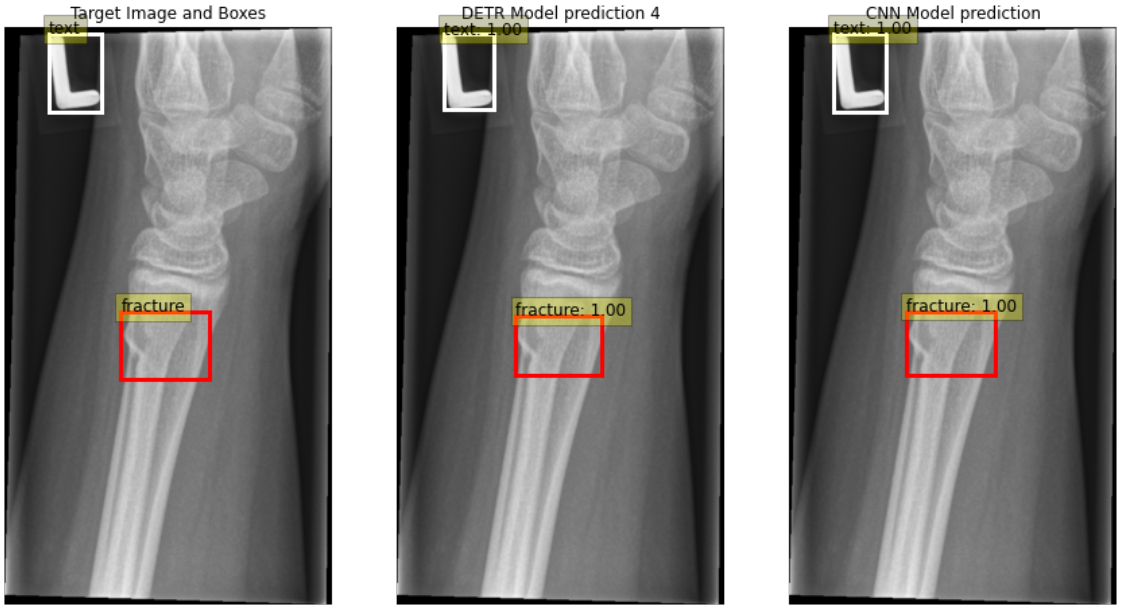


Figure 5.6: Example of detection on not-obvious Fracture

Figure 5.7: Correct detection with high confidence

The focus will be on the two best-performing models Faster RCNN and DETR with augmentation. In all the examples that will follow the first image will be the ground truth with the correct bounding boxes and classes presented, the second will host the results of the DETR model and the last will be the result of the FRCNN model. The 'Fracture' class is colored in red and the 'Text' in white.

In the first presented Figure 5.7 the reader can observe both models to accurately predict the existing fractures in cases where the fracture is obvious 5.5 and in cases when this is not happening 5.6.

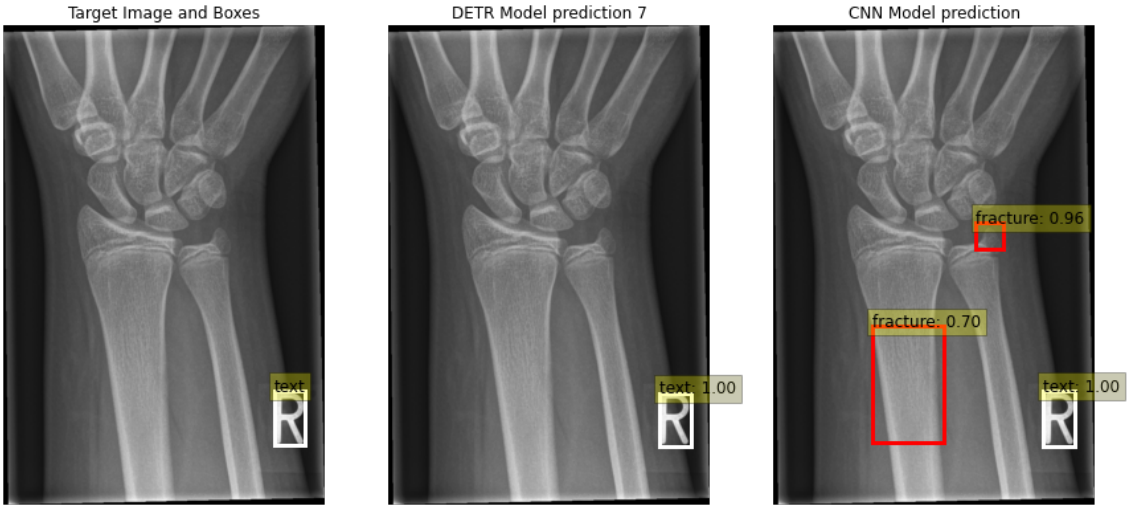


Figure 5.8: Example of FRCNN False Positive

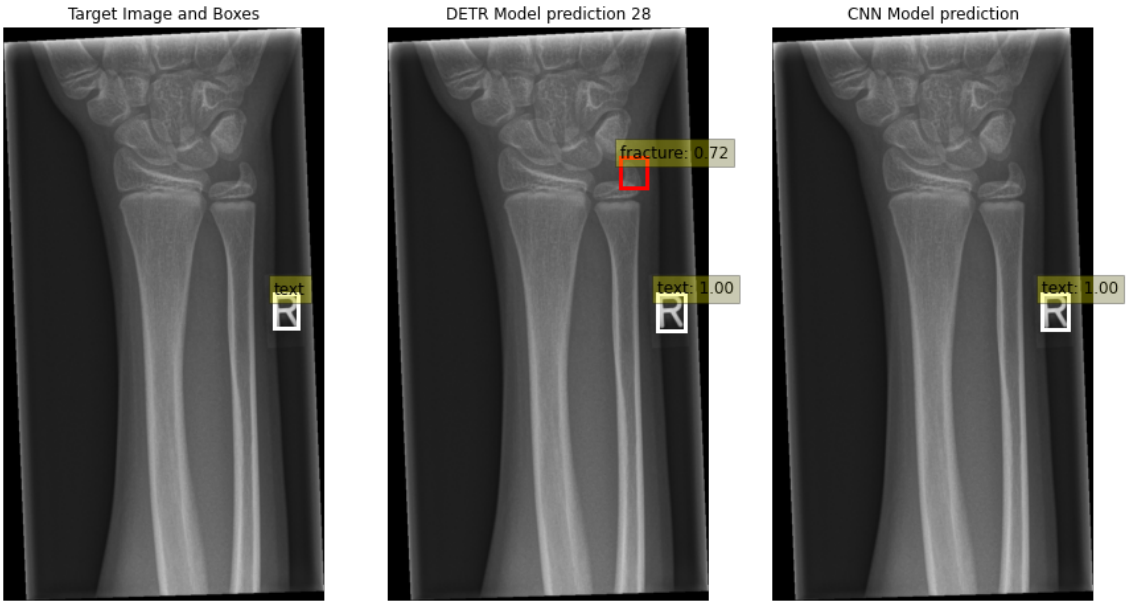


Figure 5.9: Example of DETR False Positive

Figure 5.10: Correct detection with high confidence

In the second set of images in Figure 5.10 examples of models' False Positive results are presented. In both figures, it is obvious how accurate the models are with predicting the 'Text' class, it can also be observed that when the 'Fracture' do exist their confidence is almost 1 in the presented cases, even if the fracture is not obvious, but also in many others that were not included. On the other hand, this is not the case with their False Positive prediction where they usually predict those with low confidence. This is the main reason that as shown in the pipeline it makes sense to filter the results based on the confidence and eliminate all boxes predicted with a confidence lower than 0.7.

In the following example, the value of NMS algorithm is presented. In Figure 5.11

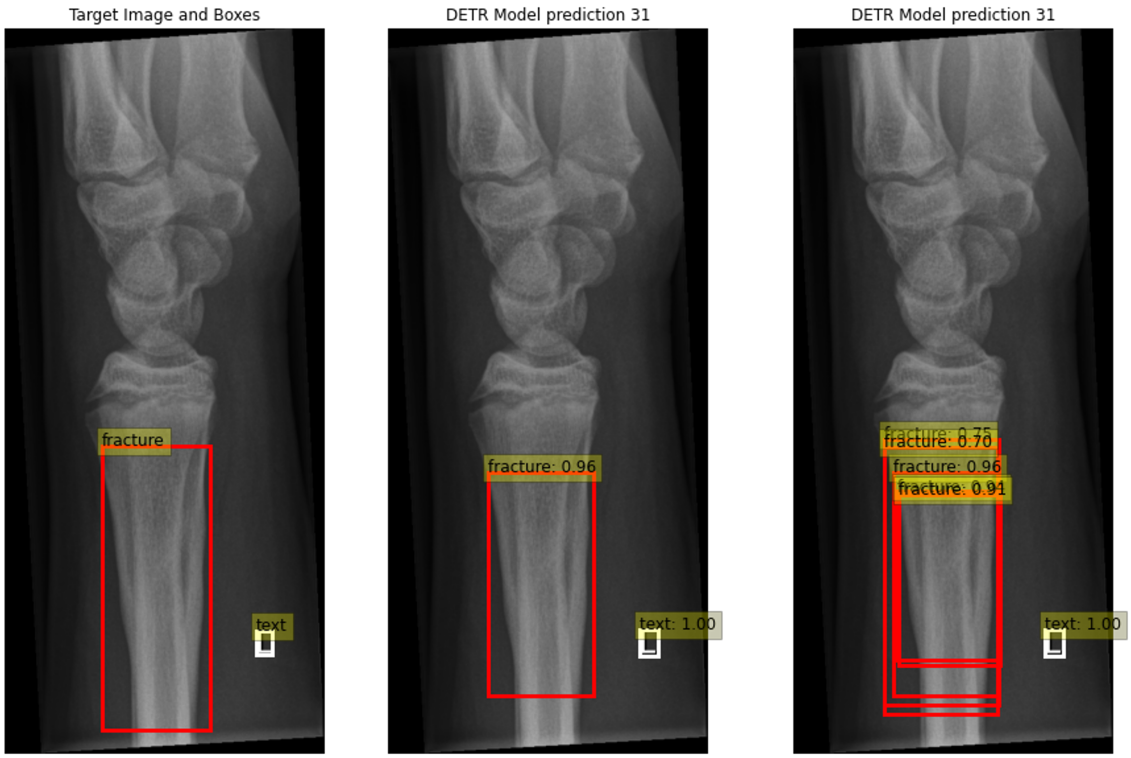


Figure 5.11: The value of NMS algorithm

In this example, it is presented an example where the same model on the same image over-predict the existing fracture. This is something common in object detection. If the Non-Maximum Suppression algorithm is used in these results then the model would have been accused of more than 4 False positives predictions lowering its AP value. The NMS algorithm eliminates all these overlapping boxes keeping the one with the highest confidence which in this specific example was a box with 96% confidence.

The last presented Figure 5.12 opens a conversation around the value of the metrics in Object detection and in this thesis in particular.

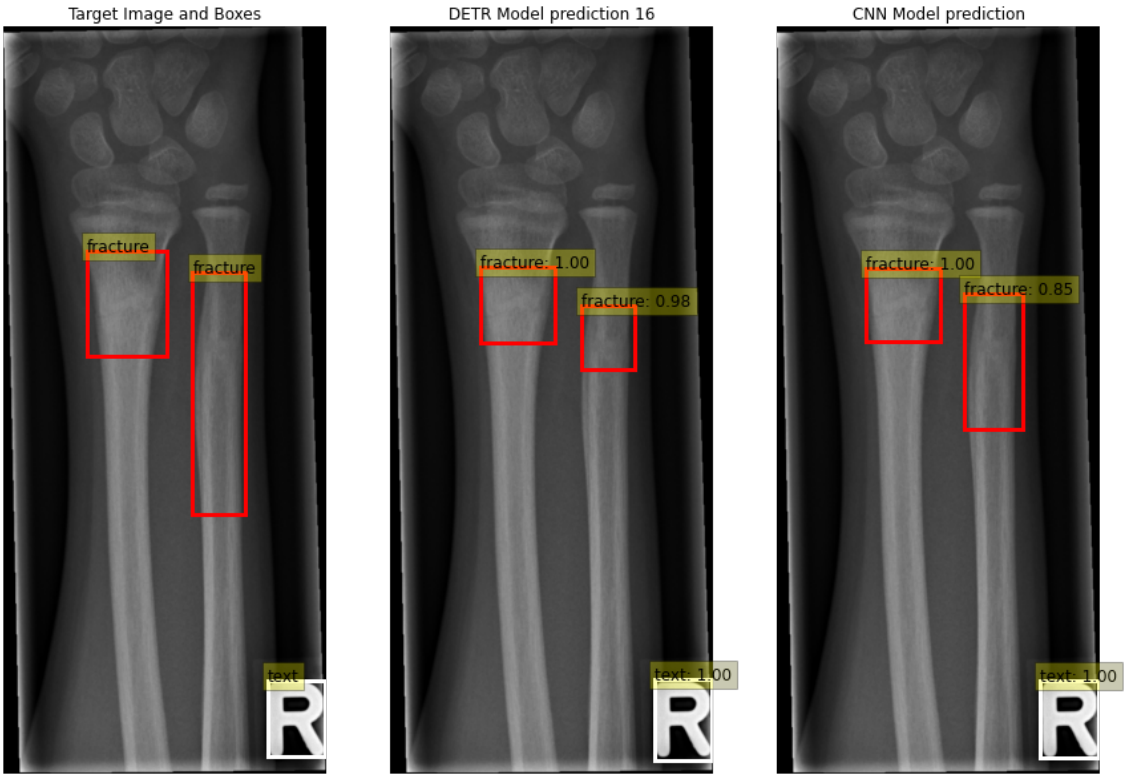


Figure 5.12: False Positives ?

Both models successfully predicted the fracture on the left bone and identified the presence of text. However, the focus of the discussion will be on the second fracture on the right side. In accordance with the previously mentioned criteria, an Intersection over Union (IoU) threshold of 0.5 was used, indicating that for a box to be considered a true positive, its IoU with the ground truth should be 50% or higher. While both models correctly detected the fracture in the right bone, the box regression was not perfect, resulting in two boxes with an IoU of less than 0.5. This implies that both models generated false positive boxes instead of true positive ones. This explains to some extent the reason behind the choice of 0.5 as a threshold in IoU and not a higher value but also generates a question on how strict these metrics are for the model. It can be argued that doctors or radiologists would be able to identify fractures in the image using either of these models with the false positive boxes they generate. The purpose of these tools is to assist the doctor in identifying the area of interest rather than precisely pinpointing the exact location of the fracture. In clinical diagnosis, the added value lies in correctly identifying the presence of a fracture rather than the precise localization. Therefore, while higher IoU thresholds may provide stricter criteria for localization accuracy, they may not significantly enhance the diagnostic value in this context.

5.1.4.1 Conclusions

Upon visual inspection of the models' performance, it becomes challenging to determine a substantial difference between them, as both models exhibit both accurate and inaccurate predictions. However, the metrics clearly indicate that the Faster RCNN model outperforms the other, contradicting the initial hypothesis that a context-aware model might surpass a model established in this field. Despite the subjective nature of visual assessment, the objective metrics provide evidence that the Faster RCNN model is the superior performer. There could be multiple reasons behind these results, and a few possible ex-

planations revolve around the dataset size and the nature of the target task. Transformer models typically require large and diverse datasets to showcase their full potential, as evidenced by the improvements achieved through data augmentation and model conversions. In this case, the transformer model took over 400 epochs to converge, resulting in a training time of 3-4 days. On the other hand, the Faster RCNN model achieved convergence after only 10 epochs, with a training time of a few hours. Moreover, in the transformer architecture, there is a layer where the images are resized between a higher accepted value of 1333 pixels and a lower accepted value of 800 pixels. This can affect the model performance in cases where the images are resized since this means that lower-quality images will be fitted into the model. Most of the images used in this dataset are in between these limits but of course, they do exist bigger and smaller images. These observations strongly suggest that fracture detection might be a domain where the Faster RCNN model outperforms transformer models. This could be attributed to the relative simplicity of the fracture detection task compared to other object detection tasks where they can be much more complex problems where the FRCCN performs poorly and the transformer models show their true potential like their performance in the COCO dataset [81].

5.2 Multi-view approach results

The adoption of a multi-view approach marks a notable change from the initial single-view methodology. The multi-view approach now presents results on a study-level basis rather than on a per-picture basis. However, this adjustment does not impact the method by which the metrics are evaluated. Despite the images being combined into a larger image with multiple bounding boxes, the fundamental process remains unchanged. The input still consists of an image, even though larger and with an increased number of boxes, and the output remains a set of proposed bounding boxes for this specific image. Consequently, the results obtained from the multi-view approach can be directly compared with those of the previous methodology.

As previously emphasized, the primary focus will be on evaluating the AP@50 value. The models that exhibit the highest performance will be presented, followed by a comprehensive discussion centered around the respective models. Due to time constraints, the augmentation pipeline was not implemented in the multiview approach. As a result, all models were trained using non-augmented data. Consequently, any comparison made will focus on the non-augmented models from both approaches.

5.2.1 Results

The model's results are presented in Table 5.5. The 'Text' label is also excluded here for the same reasons discussed before. The models achieve high values of AP in this label reaching 98% for the Faster RCNN and 92% for the DETR model. The Table presents the AP@50 value for the 'Fracture' and the mAP@50.

Model	AP@50 (Fracture)	mAP@50
DETR	0.6254	0.821
F-RCNN	**0.8776**	**0.933**

Table 5.4: Model Performance Comparison on 'Fracture' detection

The results obtained from the multiview approach provide a clear indication that the Faster RCNN model outperforms the DETR model in all metrics. The difference is almost 25%

more for the FRCNN model in the AP value for the Fracture class. What is also worth mentioning in this point is that the FRCNN results are almost the same as the ones presented in the single view approach but this will be discussed more later in the chapter.

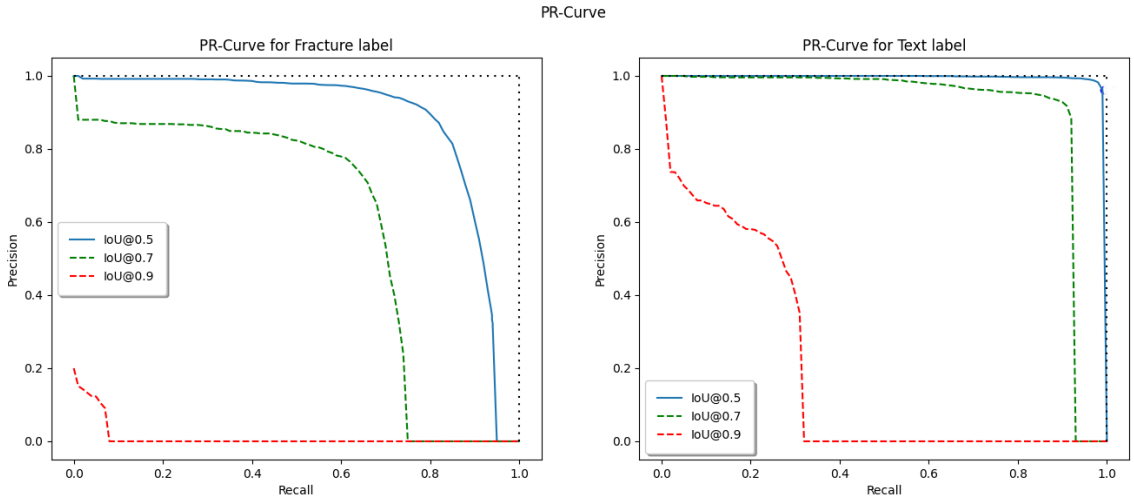


Figure 5.13: Precision-Recall curve of FRCNN

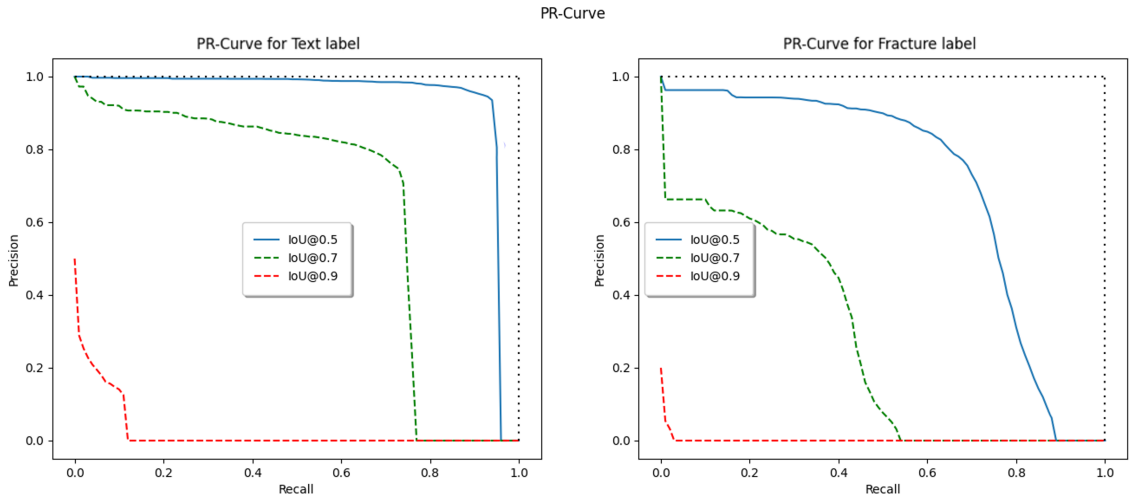


Figure 5.14: Precision-Recall curve of multiview DETR model

The Presicion-Recall curves for the two models are also provided here indicating the big difference of the model visually. The most noticeable distinction lies in how the two models respond to lower recall values. As mentioned earlier, once again, the F-RCNN model demonstrates its exceptional performance in lower sensitivities. Numerous models produced by F-RCNN achieve a precision of 1, indicating that they accurately predict every existing fracture in the test set. On the other hand, the DETR curves exhibit an instant drop in precision to around 90%, suggesting that even when the model is allowed to overpredict, it fails to detect most of the existing fractures, implying a random prediction of bounding boxes. The fracture curve of the F-RCNN closely resembles the one presented in Figure 5.1 for the non-augmented Faster-RCNN single-view model, highlighting the similarity in results between these models. Further details on the comparison between

the two approaches will be provided in the dedicated comparison section.

5.2.2 Comparison

The difference in this approach between the two models is bigger compared with the single view approach where the FRCNN also outperformed the transformer model. A total 25% difference exists in the AP value and a lower 11% in the mAP value between the models pointing out the superiority of the CNN-based model in all aspects and approaches, this is also depicted in the results for the 'Text' class where even though the DETR model didn't perform purely it had substantially lower results compared to the single view DETR model and the multi-view FRCNN. The architecture limit of the DETR model to accept images of higher resolution without resizing them seems to affect more this aspect of the problem since almost every study is going to be resized in the multiview approach. This highlights the advance of the CNN networks in this use case and points out that the DETR architecture is not yet ready to be tested in cases like this one.

5.2.3 Visual Results and Discussion

As before a discussion around the results will take place but before that, some visual examples of the model's outputs are essential for the reader as in the previous approach.

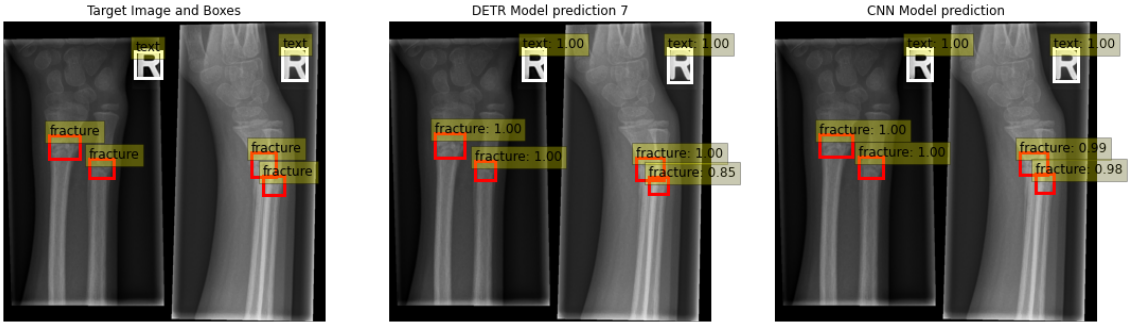


Figure 5.15: Models correct predictions

Despite the worse performance of the DETR multi-view model compare to single-view there are still many examples where the model performance was sufficient since it still managed to predict correctly many of the existing fractures. The same goes for the FR-CNN model and in the first example Figure 5.15, a very accurate prediction for both models is presented.

The following two results where the models predict a False positive are presented. For the DETR models the wrong prediction was in a picture Figure 5.17 where there was only one image for the patient. In contrast the other example Figure 5.16 shows a FRCNN false positive prediction where there were 2 images in the study. It is noteworthy to mention that the models often exhibited distinct patterns of errors. In the provided examples, it is evident that when one model made a mistake, the other correctly predicted the outcome. This observation highlights the disparities in their training and suggests that they may be focusing on different aspects of the problem at hand. However, it is important to note that proving this hypothesis is challenging, as it is solely based on visual testing and requires further investigation and analysis.

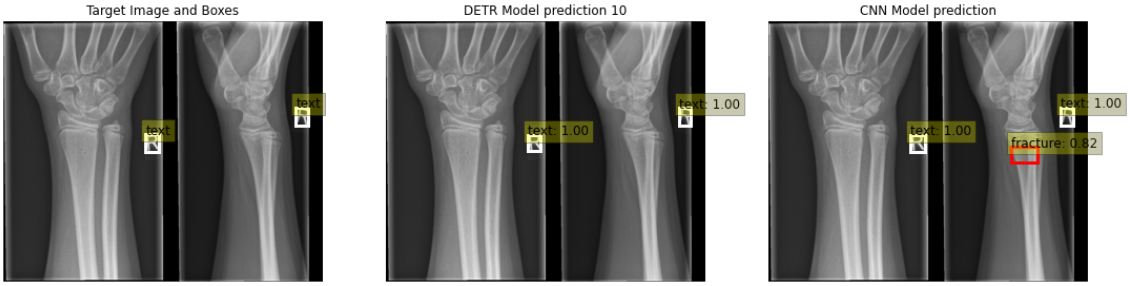


Figure 5.16: Example of FRCNN False Positive

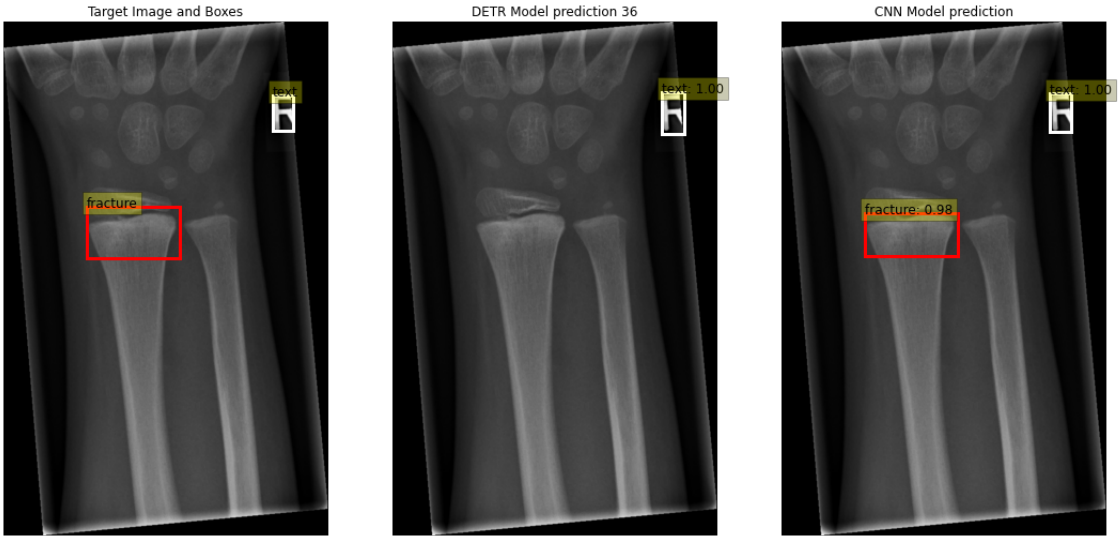


Figure 5.17: Example of DETR False Positive

The following two examples provide a clear illustration of the metrics related to Precision and Recall. In the case of the FRCNN model, it demonstrates a high Recall by successfully identifying the majority of fractures. However, this high Recall comes at the cost of a high False positives rate, resulting in lower Precision. On the other hand, the DETR model exhibits the same Precision as the FRCNN model. However, the DETR model struggles with a lower Recall, as it fails to detect all fractures present in the images. This demonstrates how a confidence score can affect the model's outcome. It is also worth mentioning at this point that the DETR model tends to predict a lot of False positives in this use case compared to the visual results in the single view.

Figure 5.18: DETR missing existing Fracture

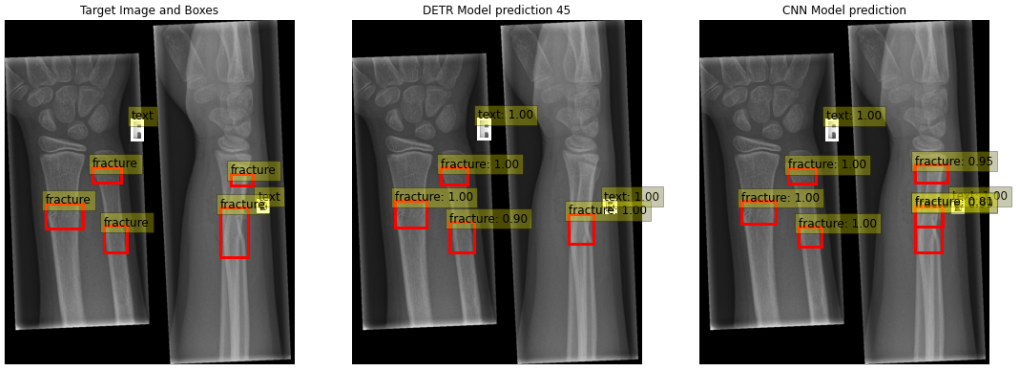


Figure 5.19: FRCNN overpredicts in the same Fracture

Last but not least, the problem discussed in the previous section on how strictly the metrics are used also exists in the multiview approach. The following image in Figure 5.20 is the exact same image as the one presented in the previous section Figure 5.12. The excess information in the second image of the study didn't seem to affect the models' prediction since they still create a smaller box for the fracture resulting in a False positive prediction. Someone would argue that a higher threshold in the NMS algorithm would alleviate this issue but what needs to be considered is that there are many cases where boxes are overlapped because of two fractures present close to each other. So more or less this solution would probably lead to lower the model's Recall since 0.7 is already a threshold that is considered high. More comparative results will be presented in the next chapter since it's quite interesting to identify how the new approach affects the model's previous decisions.

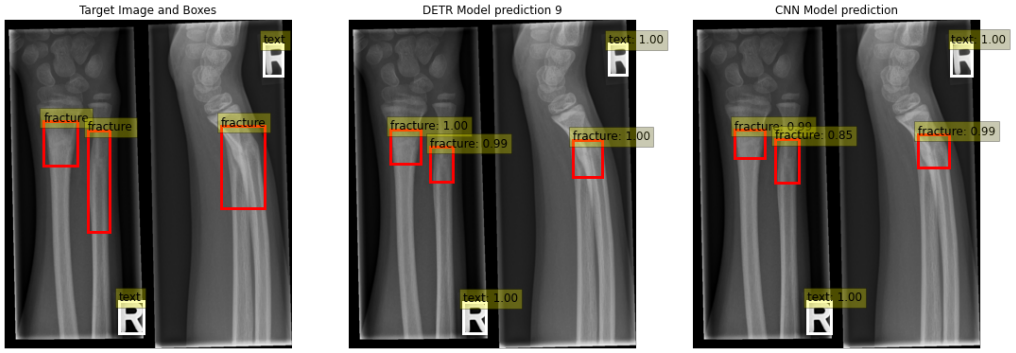


Figure 5.20: False Positives ?

The limitations and considerations discussed in the previous chapter also apply to the multiview aspect of the problem. It is important to note that the FRCNN model has been widely utilized and refined for nearly a decade, establishing its efficiency and effectiveness in various tasks. On the other hand, the DETR approach is relatively new, with only two years of existence and testing.

The superiority of the FRCNN in tasks that have proven its efficiency will be difficult to be trampled upon, refuting the initial thesis hypothesis that a context-aware object detection model, such as DETR, could outperform FRCNN in use cases like fracture detection.

The potential difficulty lies in competing against a well-established model that has been optimized and fine-tuned over many years. While the promise of a context-aware approach is compelling, it will require substantial evidence and advancements to convincingly demonstrate its superiority over the FRCNN model in this particular application.

5.3 Approach comparison and discussion

In this chapter, a comparison between the two approaches will be performed. As mentioned before since an Augmentation pipeline was not implemented in the multiview approach the two models to compare will be the Faster RCNNs without augmentation from each approach.

Model	AP@50 (Fracture)	mAP@50
F-RCNN single view	**0.8815**	**0.935**
F-RCNN multi-view	0.8776	0.933

Table 5.5: Single vs Multiview Performance Comparison

Both models exhibit nearly identical results, highlighting the potential insignificance of their differences when considering the average values obtained from multiple runs. The Faster RCNN model continues to demonstrate excellent performance, even in a more complex problem setting characterized by larger images and a reduced training dataset size.

The challenge in the multiview problem lies in the context awareness it demands. In this scenario, the overall context of the image significantly influences the detection results. Consequently, correlating two pixels that exist in different images becomes more challenging. The larger image size and the limited training data further amplify the complexity of the multiview problem. With larger images, the model needs to process more pixels, requiring additional computational resources and increasing training time. The shortage of training data poses another obstacle, as it becomes more difficult to establish robust correlations and patterns across multiple views.

Considering these factors, the Faster RCNN model's ability to achieve impressive results in this context-aware, multiview problem is a testament to its robustness and effectiveness. Furthermore, the multiview model offers the advantage of simultaneously predicting an entire study, enclosing multiple views, while still maintaining the capability to predict individual single-view images. This added flexibility provides doctors with a valuable decision-making tool that allows them to choose the most suitable approach based on their specific needs and proceed accordingly, enabling them to adapt their approach based on the complexity of the case, time constraints, and the desired level of analysis. It empowers them to make informed decisions and optimize their diagnostic processes for improved patient care.

Additionally, upon visual inspection, there were instances where the two approaches yielded different decisions, indicating that the multiview approach had an impact on the training process. The utilization of multiple views influenced the model's learning and predictive abilities, leading to divergent outcomes in certain cases. An example of that is demonstrated in Figure 5.23

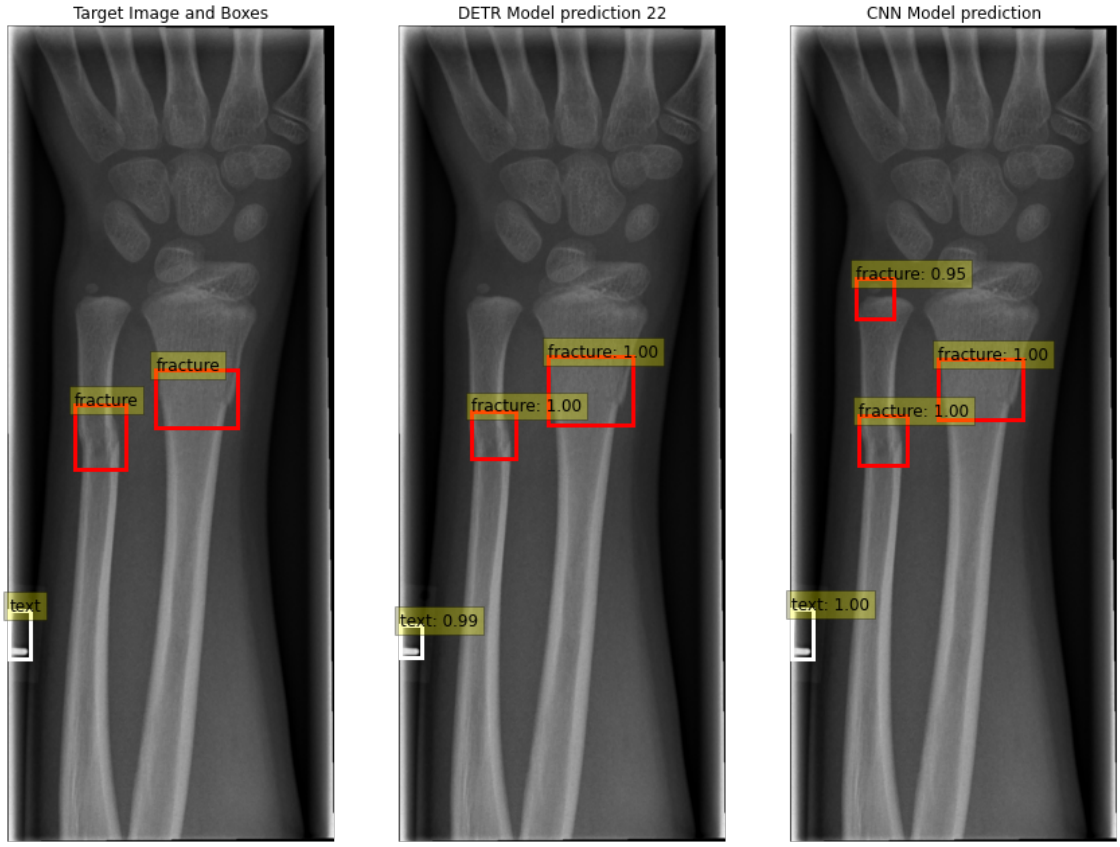


Figure 5.21: Example of singleview prediction

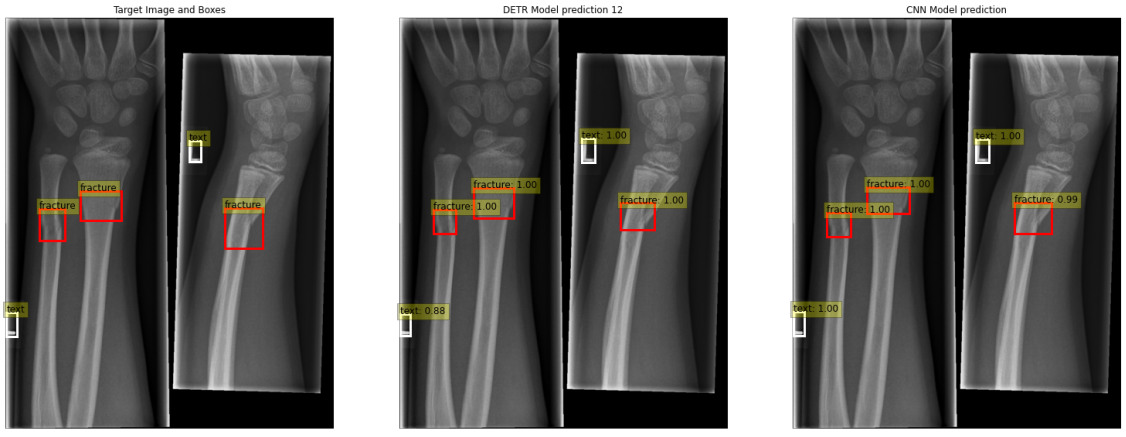


Figure 5.22: Example of multiview prediction

Figure 5.23: Visual comparison of the models predictions

In the first image of Figure 5.21, the FRCNN model predicted a false positive fracture at the top with a high confidence of 95%. However, this fracture was not detected in the multiview prediction. It is important to note that due to the limited data available, it is difficult to draw definitive conclusions or make safe assumptions about the impact of the multiview approach on the overall outcome. Although the AP values suggest that the models are similar, these specific examples suggest potential differences in the training process. Further investigation is needed to better understand the influence of the multiview approach.

In conclusion, the comparison between the multiview and single-view models reveals interesting insights. Despite the nearly identical results obtained by both models, the multiview approach offers distinct advantages. The Faster RCNN model demonstrates exceptional performance in both single and multiview problems, tackling the complexities posed by larger images and a reduced training dataset size.

6 Conclusion

This thesis has dived into the investigation of fracture detection using deep learning models, employing various model architectures and different approaches. The primary objective was to develop a robust suggestion tool capable of assisting doctors in enhancing their accuracy while saving valuable time. An initial hypothesis regarding the effectiveness of a ”Context Awareness” model in the practical application of fracture detection was stated in the beginning. The main question was whether a model capable of understanding the entire image would outperform traditional methods in identifying fractures, as this requires considering the complete image information to distinguish fractures from other bone abnormalities. This hypothesis lead to a study that involved a comparison between two distinct model architectures: the Faster RCNN, renowned for its achievements in object detection, and the Detection Transformer, a state-of-the-art architecture that incorporates global attention layers, enabling each region of interest to consider the entire image context. To facilitate a fair and meaningful comparison, a dedicated Wrist-only fracture detection dataset was utilized throughout the experiments. By maintaining consistency in data preprocessing, evaluation, and experimental procedures, the aim was to ascertain definitive results that could either confirm or refute the initial hypothesis.

The results of the experiments demonstrated that the Faster RCNN model outperformed the Detection Transformer model, thereby disproving the initial hypothesis. The Faster RCNN achieved an impressive AP@50 value of 89.9%, whereas the Detection Transformer achieved 78%. This significant difference in performance suggests that, in the context of fracture detection, the Faster RCNN model is better suited for the task at hand.

Based on these findings, a second hypothesis was formulated and investigated in relation to the primary objective. When analyzing fracture datasets, the data is typically categorized into patients and studies. A study consists of multiple X-ray images of the same hand taken from different perspectives, which aids doctors in gaining a comprehensive understanding of the patient’s hand structure and facilitates more accurate fracture detection. In light of this context, experiments were conducted to explore how this information could be effectively integrated into deep learning models. The proposed solution involved merging all the images within a study into a single, large image that includes all the bounding boxes. This merged image was then fed into the model as a unified input. By doing so, the model could access the entirety of the available information for each patient, mimicking the way a doctor examines the entire study rather than focusing on individual images. This approach holds the potential to enable the model to make more informed decisions by considering the complete study rather than relying solely on one image.

To ensure a fair comparison, the experiments utilized the same model and architectures as the previous study. The Faster RCNN model once again demonstrated superior performance over the DETR model in the context of multiview analysis, achieving an AP@50 value of 87.8% compared to 62.5% for DETR. These consistent results with the Faster RCNN model left the hypothesis that a multiview approach could enhance the efficacy of suggested models open, as it achieved similar performance to the single view approach dealing with a more complex problem. The value of this approach increases when it is considered that these models can predict a whole study and a single image.

Conversely, the fact that the DETR model did not surpass the Faster RCNN model in this approach revealed certain limitations compared to CNN architectures. These limitations

can be summarized as follows: architecture constraints due to image resizing, diminished performance when working with limited data volumes, and a lack of fine-tuning since the DETR model had only been introduced one year prior. Nonetheless, the current results indicate the possibility that DETR could potentially compete with other well-established models in the future, given different datasets, alternative approaches, and a more refined model through fine-tuning.

It is crucial to acknowledge the limitations of the presented approaches and models and identify potential areas for future research. One limitation of this project that prevents the generalization of the results is the use of a specific dataset consisting of pediatric wrist X-rays. This restricts the models' ability to predict other body parts or patients with a wider age distribution. Fine-tuning the models with more diverse data is essential to validate the results and ensure broader applicability. This can be a focus of future projects. Additionally, due to time constraints, the resizing layers in the DETR model had an impact on training, and no specific strategy was formulated to overcome or address this issue. Particularly in the comparison between the DETR single-view and DETR multi-view models, the fairness of the comparison is compromised, as indicated by the results. A more equitable approach would involve resizing the single-view images with the same aspect ratio as the multi-view images. Lastly, the limited time available for hyperparameter tuning restricted the models' performance optimization. Further work in this area could lead to even better results.

In conclusion, this thesis has investigated the hypothesis of employing a ”Context Awareness” model for fracture detection using two different approaches, aiming to create a powerful suggestion tool for doctors. The comparison between the Faster RCNN and Detection Transformer architectures has provided valuable insights into their respective strengths and limitations. The findings of this study contribute to the advancement of fracture detection methodologies and lay the groundwork for further research and development in this critical field of medical imaging.

Bibliography

[1] Jun Deng et al. “A review of research on object detection based on deep learning”. en. In: *Journal of Physics: Conference Series* 1684.1 (Nov. 2020), p. 012028. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1684/1/012028. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1684/1/012028.

[2] Ai-Min Wu et al. “Global, regional, and national burden of bone fractures in 204 countries and territories, 1990–2019: a systematic analysis from the Global Burden of Disease Study 2019”. English. In: *The Lancet Healthy Longevity* 2.9 (Sept. 2021). Publisher: Elsevier, e580–e592. ISSN: 2666-7568. DOI: 10.1016/S2666-7568(21)00172-0. URL: https://www.thelancet.com/journals/lanhl/article/PIIS2666-7568(21)00172-0/fulltext.

[3] Mark J. Halsted et al. “Diagnostic errors by radiology residents in interpreting pediatric radiographs in an emergency setting”. eng. In: *Pediatric Radiology* 34.4 (Apr. 2004), pp. 331–336. ISSN: 0301-0449. DOI: 10.1007/s00247-004-1150-7.

[4] *Articles*. en-US. URL: https://www.cedars-sinai.org/health-library/articles.html (visited on 05/18/2023).

[5] *X-Rays*. eng. Text. Publisher: National Library of Medicine. URL: https://medlineplus.gov/xrays.html (visited on 05/18/2023).

[6] P. Viola and M. Jones. “Rapid object detection using a boosted cascade of simple features”. en. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. Kauai, HI, USA: IEEE Comput. Soc, 2001, pp. I–511–I–518. ISBN: 978-0-7695-1272-3. DOI: 10.1109/CVPR.2001.990517. URL: http://ieeexplore.ieee.org/document/990517/ (visited on 05/29/2023).

[7] N. Dalal and B. Triggs. “Histograms of Oriented Gradients for Human Detection”. en. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 886–893. ISBN: 978-0-7695-2372-9. DOI: 10.1109/CVPR.2005.177. URL: http://ieeexplore.ieee.org/document/1467360/ (visited on 05/29/2023).

[8] Zhengxia Zou et al. *Object Detection in 20 Years: A Survey*. arXiv:1905.05055 [cs]. Jan. 2023. URL: http://arxiv.org/abs/1905.05055 (visited on 05/29/2023).

[9] Chinmoy Borah. *Evolution of Object Detection*. en. Nov. 2020. URL: https://medium.com/analytics-vidhya/evolution-of-object-detection-582259d2aa9b (visited on 05/29/2023).

[10] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. arXiv:2005.12872 [cs]. May 2020. URL: http://arxiv.org/abs/2005.12872 (visited on 05/29/2023).

[11] *(3) (PDF) Semantic Image Cropping*. URL: https://www.researchgate.net/publication/353284602_Semantic_Image_Cropping/figures?lo=1 (visited on 06/08/2023).

[12] *Papers with Code - 3D Object Detection*. en. URL: https://paperswithcode.com/task/3d-object-detection (visited on 05/29/2023).

[13] *Universal Image Segmentation with Mask2Former and OneFormer*. URL: https://huggingface.co/blog/mask2former (visited on 05/29/2023).

[14] *Papers with Code - Multi-Object Tracking*. en. URL: https://paperswithcode.com/task/multi-object-tracking (visited on 05/29/2023).

[15] Bin Guan et al. “Thigh fracture detection using deep learning method based on new dilated convolutional feature pyramid network”. en. In: *Pattern Recognition Letters* 125 (July 2019), pp. 521–526. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2019.06.015.

URL: https://www.sciencedirect.com/science/article/pii/S0167865519301825 (visited on 05/31/2023).

[16] Bin Guan et al. “Arm fracture detection in X-rays based on improved deep convolutional neural network”. en. In: *Computers & Electrical Engineering* 81 (Jan. 2020), p. 106530. ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2019.106530. URL: https://www.sciencedirect.com/science/article/pii/S0045790618330878 (visited on 05/31/2023).

[17] Mengxuan Wang et al. “ParallelNet: multiple backbone network for detection tasks on thigh bone fracture”. en. In: *Multimedia Systems* 27.6 (Dec. 2021), pp. 1091–1100. ISSN: 1432-1882. DOI: 10.1007/s00530-021-00783-9. URL: https://doi.org/10.1007/s00530-021-00783-9 (visited on 05/31/2023).

[18] Yangling Ma and Yixin Luo. “Bone fracture detection through the two-stage system of Crack-Sensitive Convolutional Neural Network”. en. In: *Informatics in Medicine Unlocked* 22 (Jan. 2021), p. 100452. ISSN: 2352-9148. DOI: 10.1016/j.imu.2020.100452. URL: https://www.sciencedirect.com/science/article/pii/S235291482030602X (visited on 05/31/2023).

[19] Gang Sha, Junsheng Wu, and Bin Yu. *Detection of Spinal Fracture Lesions based on Improved Yolov2*. Pages: 238. June 2020. DOI: 10.1109/ICAICA50127.2020.9182582.

[20] Gang Sha, Junsheng Wu, and Bin Yu. “The improved faster-RCNN for spinal fracture lesions detection”. In: *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology* 42.6 (Jan. 2022), pp. 5823–5837. ISSN: 1064-1246. DOI: 10.3233/JIFS-212389. URL: https://doi.org/10.3233/JIFS-212389 (visited on 05/31/2023).

[21] Linyan Xue et al. “Detection and localization of hand fractures based on GA_Faster R-CNN”. en. In: *Alexandria Engineering Journal* 60.5 (Oct. 2021), pp. 4555–4562. ISSN: 1110-0168. DOI: 10.1016/j.aej.2021.03.005. URL: https://www.sciencedirect.com/science/article/pii/S1110016821001605 (visited on 05/31/2023).

[22] Eszter Nagy et al. “A pediatric wrist trauma X-ray dataset (GRAZPEDWRI-DX) for machine learning”. en. In: *Scientific Data* 9.1 (May 2022). Number: 1 Publisher: Nature Publishing Group, p. 222. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01328-z. URL: https://www.nature.com/articles/s41597-022-01328-z.

[23] Thorsten Franzel, Uwe Schmidt, and Stefan Roth. *Object Detection in Multi-view X-Ray Images*. Aug. 2012. ISBN: 978-3-642-32716-2. DOI: 10.1007/978-3-642-32717-9_15.

[24] Brian K. S. Isaac-Medina, Chris G. Willcocks, and Toby P. Breckon. “Multi-view Object Detection Using Epipolar Constraints within Cluttered X-ray Security Imagery”. en. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. Milan, Italy: IEEE, Jan. 2021, pp. 9889–9896. ISBN: 978-1-72818-808-9. DOI: 10.1109/ICPR48806.2021.9413007. URL: https://ieeexplore.ieee.org/document/9413007/ (visited on 05/31/2023).

[25] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv:1506.01497 [cs]. Jan. 2016. DOI: 10.48550/arXiv.1506.01497. URL: http://arxiv.org/abs/1506.01497 (visited on 05/31/2023).

[26] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. arXiv:1311.2524 [cs]. Oct. 2014. URL: http://arxiv.org/abs/1311.2524 (visited on 05/31/2023).

[27] Yugesh Verma. *R-CNN vs Fast R-CNN vs Faster R-CNN - A Comparative Guide*. en-US. Sept. 2021. URL: https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/ (visited on 05/31/2023).

[28] Sik-Ho Tsang. *Review: Faster R-CNN (Object Detection)*. en. Mar. 2019. URL: https://towardsdatascience.com/review-faster-r-cnn-object-detection-f5685cb30202 (visited on 05/31/2023).
[29] Ross Girshick. *Fast R-CNN*. arXiv:1504.08083 [cs]. Sept. 2015. URL: http://arxiv.org/abs/1504.08083 (visited on 05/31/2023).
[30] *2) Fast R-CNN*. ko. URL: https://wikidocs.net/148634 (visited on 05/31/2023).
[31] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Dec. 2017. URL: http://arxiv.org/abs/1706.03762 (visited on 06/02/2023).
[32] Eduardo Muñoz. *Attention is all you need: Discovering the Transformer paper*. en. Feb. 2021. URL: https://towardsdatascience.com/attention-is-all-you-need-discovering-the-transformer-paper-73e5ff5e0634 (visited on 06/02/2023).
[33] *Transformers from scratch | peterbloem.nl*. URL: https://peterbloem.nl/blog/transformers (visited on 06/04/2023).
[34] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929 [cs]. June 2021. URL: http://arxiv.org/abs/2010.11929 (visited on 06/02/2023).
[35] Adem Akdogan. *DETR: End-to-End Object Detection with Transformers and Implementation of Python*. en. Aug. 2022. URL: https://towardsdatascience.com/detr-end-to-end-object-detection-with-transformers-and-implementation-of-python-8f195015c94d (visited on 06/02/2023).
[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.
[37] Jason Brownlee. *A Gentle Introduction to Transfer Learning for Deep Learning*. en-US. Dec. 2017. URL: https://machinelearningmastery.com/transfer-learning-for-deep-learning/ (visited on 05/21/2023).
[38] *General Data Protection Regulation (GDPR) – Official Legal Text*. en-US. URL: https://gdpr-info.eu/ (visited on 05/21/2023).
[39] Alexander Ke et al. "CheXtransfer: performance and parameter efficiency of ImageNet models for chest X-Ray interpretation". In: *Proceedings of the Conference on Health, Inference, and Learning*. CHIL '21. New York, NY, USA: Association for Computing Machinery, Apr. 2021, pp. 116–124. ISBN: 978-1-4503-8359-2. DOI: 10.1145/3450439.3451867. URL: https://dl.acm.org/doi/10.1145/3450439.3451867 (visited on 05/21/2023).
[40] *ImageNet*. URL: https://www.image-net.org/.
[41] *Faster R-CNN — Torchvision main documentation*. URL: https://pytorch.org/vision/main/models/faster_rcnn.html (visited on 05/21/2023).
[42] *COCO - Common Objects in Context*. URL: https://cocodataset.org/#detection-eval (visited on 05/27/2023).
[43] *COCO - Common Objects in Context*. URL: https://cocodataset.org/#home.
[44] *Evaluating Object Detection Models: Guide to Performance Metrics*. en. Oct. 2019. URL: https://manalelaidouni.github.io/manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html (visited on 05/23/2023).
[45] baeldung. *Intersection Over Union for Object Detection | Baeldung on Computer Science*. en-US. Apr. 2022. URL: https://www.baeldung.com/cs/object-detection-intersection-vs-union (visited on 05/23/2023).
[46] *Evaluation Metrics for Object Detection*. en-US. Aug. 2020. URL: https://debuggercafe.com/evaluation-metrics-for-object-detection/ (visited on 05/23/2023).
[47] *Object Detection: Calculating mean Average Precision (mAP) with Confidence - KiKaBeN*. en-US. May 2022. URL: https://kikaben.com/object-detection-mean-average-precision/ (visited on 05/23/2023).

[48] *Evaluating Object Detection Models: Guide to Performance Metrics*. en. Oct. 2019. URL: https://manalelaidouni.github.io/manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html (visited on 05/25/2023).
[49] Aqeel Anwar. *What is Average Precision in Object Detection & Localization Algorithms and how to calculate it?* en. May 2022. URL: https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b (visited on 06/08/2023).
[50] Vineeth S. Subramanyam. *Non Max Suppression (NMS)*. en. Jan. 2021. URL: https://medium.com/analytics-vidhya/non-max-suppression-nms-6623e6572536 (visited on 05/28/2023).
[51] Dimitrios Papadopoulos. *Object Detection*. University Lecture. 2022.
[52] Sambasivarao K. *Non-maximum Suppression (NMS)*. en. Apr. 2021. URL: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c (visited on 05/28/2023).
[53] *cocodataset/cocoapi*. original-date: 2015-01-25T20:26:39Z. May 2023. URL: https://github.com/cocodataset/cocoapi/blob/8c9bcc3cf640524c4c20a9c40e89cb6a2f2fa0e9/PythonAPI/pycocotools/cocoeval.py (visited on 05/27/2023).
[54] *Supervisely: unified OS for computer vision*. URL: https://supervisely.com/.
[55] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". en. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338. ISSN: 1573-1405. DOI: 10.1007/s11263-009-0275-4. URL: https://doi.org/10.1007/s11263-009-0275-4.
[56] Glenn Jocher et al. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. Oct. 2020. DOI: 10.5281/zenodo.4154370. URL: https://zenodo.org/record/4154370.
[57] K. S. Berbaum et al. "Satisfaction of search in diagnostic radiology". eng. In: *Investigative Radiology* 25.2 (Feb. 1990), pp. 133–140. ISSN: 0020-9996. DOI: 10.1097/00004424-199002000-00006.
[58] Henry Knipe. *Satisfaction of search error | Radiology Reference Article | Radiopaedia.org*. en-US. DOI: 10.53347/rID-31220. URL: https://radiopaedia.org/articles/satisfaction-of-search-error (visited on 06/08/2023).
[59] Neil C. Thompson et al. *The Computational Limits of Deep Learning*. arXiv:2007.05558 [cs, stat]. July 2022. URL: http://arxiv.org/abs/2007.05558.
[60] Renu Khandelwal. *COCO data format for Object detection*. en. Dec. 2019. URL: https://towardsdatascience.com/coco-data-format-for-object-detection-a4c5eaf518c5.
[61] *figshare - credit for all your research*. URL: https://figshare.com/.
[62] *opencv-python: Wrapper package for OpenCV python bindings.* URL: https://github.com/opencv/opencv-python.
[63] *ultralytics/yolov5 at blog.roboflow.com*. en. URL: https://github.com/ultralytics/yolov5.
[64] Haobin Tan. *Annotation Conversion: COCO JSON to YOLO Txt*. en-us. Dec. 2020. URL: https://haobin-tan.netlify.app/ai/computer-vision/object-detection/coco-json-to-yolo-txt/.
[65] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.
[66] *The Essential Guide to Data Augmentation in Deep Learning*. en. URL: https://www.v7labs.com/blog/data-augmentation-guide,%20https://www.v7labs.com/blog/data-augmentation-guide.
[67] *Albumentations*. en. URL: https://albumentations.ai/.

[68] *torchvision — Torchvision main documentation*. URL: https://pytorch.org/vision/stable/index.html.
[69] *Radiobotics — We Augment Radiology through AI & Machine Learning*. URL: https://www.radiobotics.com/.
[70] *Conda — conda documentation*. URL: https://docs.conda.io/en/latest/ (visited on 06/06/2023).
[71] *Visual Studio Code - Code Editing. Redefined*. en. URL: https://code.visualstudio.com/ (visited on 06/06/2023).
[72] *Download Python | Python.org*. URL: https://www.python.org/downloads/ (visited on 06/06/2023).
[73] *GitHub - cookiecutter/cookiecutter: A cross-platform command-line utility that creates projects from cookiecutters (project templates), e.g. Python package projects, C projects.* URL: https://github.com/cookiecutter/cookiecutter (visited on 06/06/2023).
[74] *Git*. URL: https://git-scm.com/ (visited on 06/06/2023).
[75] *GitHub: Let's build from here*. en. URL: https://github.com/ (visited on 06/06/2023).
[76] *Weights & Biases – Developer tools for ML*. URL: https://wandb.ai/site/,%20http://wandb.ai/site (visited on 06/06/2023).
[77] *FileZilla - The free FTP solution*. URL: https://filezilla-project.org/ (visited on 06/13/2023).
[78] *What is MLOps?* en-US. Dec. 2021. URL: https://www.databricks.com/glossary/mlops (visited on 06/06/2023).
[79] *ConvNet for Classifying Cifar-10 (part 1)*. en. Feb. 2019. URL: https://mydeeplearningnb.wordpress.com/2019/02/23/convnet-for-classification-of-cifar-10/ (visited on 06/06/2023).
[80] *PyTorch*. en. URL: https://www.pytorch.org (visited on 06/13/2023).
[81] *Papers with Code - COCO test-dev Benchmark (Object Detection)*. en. URL: https://paperswithcode.com/sota/object-detection-on-coco (visited on 06/11/2023).

A Appendix

In this thesis, we present additional information and supplementary materials in the form of an appendix. The purpose of the appendix is to provide supporting details, data, and documentation that are relevant to the main research presented in the body of the thesis. The appendix serves as a repository of additional material that contributes to a comprehensive understanding of the research topic. It includes materials that are too extensive or detailed to be included within the main body of the thesis but are still valuable for readers who seek more in-depth knowledge or wish to replicate and validate the research findings.

A.1 Object detection data formats

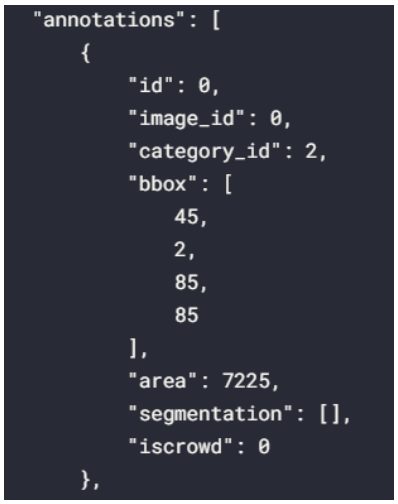
A detailed representation of the JSON file of the COCO dataset is shown in the figure below. Each one of the fields mentioned before are containing different tags. Some of these tags are mandatory to be used like the image id for example while others are not or it depends on the model that is going to be used after.

(a) Info field of the JSON file

(b) License field

(c) Category field can contain more than one

(d) Sniped code of one image of the dataset



(e) Sniped code of one annotation with one box

Figure A.1: The COCO format JSON file

The other available formats of the data will be presented along with a comparison of all of them at the end. Starting with the Pascal Visual Object Classes (PascalVOC) format the differences from the COCO format will be elaborated to better understand both formats.

- Pascal is using an XML file instead of a JSON that is used in the COCO.
- An independent XML file is saved for each image in the dataset instead of one for the whole dataset.
- The bounding box format is different from the Pascal format.
  - Pascal VOC Bbox :[xmin-top left, ymin-top left,xmax-bottom right, ymax-bottom right]

```
<annotation>
        <folder>Kangaroo</folder>
        <filename>00001.jpg</filename>
        <path>./Kangaroo/stock-12.jpg</path>
        <source>
                <database>Kangaroo</database>
        </source>
        <size>
                <width>450</width>
                <height>319</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>kangaroo</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>233</xmin>
                        <ymin>89</ymin>
                        <xmax>386</xmax>
                        <ymax>262</ymax>
                </bndbox>
        </object>
</annotation>
```

Figure A.2: Example of Pascal data

The last major format in object detection is the You Only Look Once (YOLO). The YOLO format does not include additional information such as segmentation masks or key points so usually, it only works with object detection tasks. Similar to Pascal the key differences of this format can be summarized

- Yolo saves the annotations in a TXT format file.
- One annotation file is saved for each object existing per image in the dataset. Meaning that if in 2 images there are 10 bounding boxes in total, 10 text files will be needed.
- Each line has 5 values separated by spaces: the object class index (starting from 0), the normalized x-coordinate of the center of the bounding box, the normalized y-coordinate of the center of the bounding box, the normalized width of the bounding box, and the normalized height of the bounding box.
- The YOLO bounding box structure is also different

– A bounding box is represented by four values [x_center, y_center, width, height]. x_center and y_center are the normalized coordinates of the center of the bounding box.

To give the reader a better intuition of what the differences are between the formats bounding boxes representation an example is presented in figure A.3

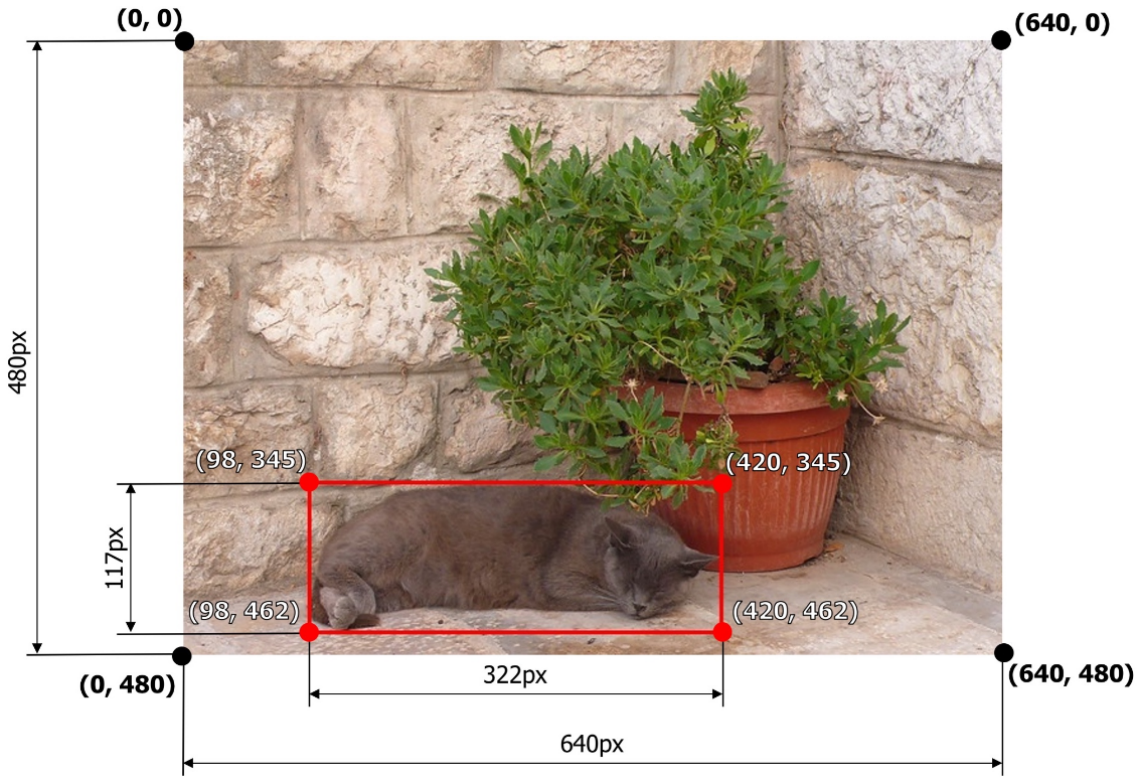


Figure A.3: Example of bounding boxes [67]

In the example picture, the bounding boxes for each format would be

- COCO : [98, 345, 322, 117]
- Pascal : [98, 345, 420, 462]
- YoLo: Coordinates of the example bounding box in this format are [((420 + 98) / 2) / 640, ((462 + 345) / 2) / 480, 322 / 640, 117 / 480] which are [0.4046875, 0.840625, 0.503125, 0.24375]

Technical
University of
Denmark

Brovej, Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

www.compute.dtu.dk/