

YEDITEPE UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DISTAL RADIUS FRACTURE DETECTION: A COMPARISON OF COMPUTER VISION ALGORITHMS

A Thesis Submitted

by

Burcu Selçuk

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in

Computer Engineering

Supervisor

Assist. Prof. Dr. Tacha Serif

Istanbul - 2025

DISTAL RADIUS FRACTURE DETECTION: A COMPARISON OF COMPUTER VISION ALGORITHMS

by

Burcu Selçuk

Approved by:

Assist. Prof. Dr. Tacha Serif
(Yeditepe University) .....
(Thesis Supervisor)

Prof. Dr. Bülent Tavlı
(TOBB University of Economics and Technology) .....

Assist. Prof. Dr. Onur Demir
(Yeditepe University) .....

DATE OF APPROVAL: .... / .... / 20....

DECLARATION OF ORIGINALITY

I hereby declare that this thesis is my own work and that all information in this thesis has been obtained and presented following academic rules and ethical conduct. I have fully cited and referenced all material and results as required by these rules and conduct, and this thesis study does not contain any plagiarism. The necessary permissions have been obtained if any material used in the thesis requires copyright. No material from this thesis has been used to award another degree.

To the best of my knowledge and belief, it contains no material previously published or written by another person nor material accepted for the award of any other degree except where due acknowledgment has been made in the text.

I accept all kinds of legal liability that may arise in cases contrary to these situations.

Burcu Selçuk

.....

ABSTRACT

DISTAL RADIUS FRACTURE DETECTION: A COMPARISON OF COMPUTER VISION ALGORITHMS

Distal radius fractures (DRFs) are among the most common skeletal injuries encountered in emergency and orthopedic settings, accounting for a significant portion of wrist-related trauma cases across all age groups. Despite their prevalence, subtle fractures are frequently missed, particularly by less experienced clinicians, due to variations in anatomical presentation and radiographic complexity. Early and accurate detection is essential for ensuring proper treatment and avoiding long-term functional impairment.

This study presents a deep learning-based system for the automatic detection and localization of DRFs in wrist radiographs, developed with the goal of supporting both clinical workflows and medical education. Accordingly, several state-of-the-art object detection and segmentation models are evaluated, including Faster R-CNN, YOLOv8, YOLOv11, Mask R-CNN, and EfficientNet. Each model is trained and validated using labeled datasets, with performance compared across precision, recall, and F1-score metrics. Based on these findings, EfficientNet emerged as the best-performing model and is subsequently deployed within a custom-designed Python application. The prototype enables users to upload radiographs, view segmentation-based predictions, and submit diagnostic feedback, making it suitable for use in both clinical and training environments.

To validate the system's diagnostic potential, an independent evaluation is conducted using 40 radiographs annotated by orthopedic surgeons. The EfficientNet model is compared against five sixth-year medical interns, each of whom manually annotated the same cases. The results show that the model outperforms the interns across all major metrics, achieving a precision of 0.89, recall of 0.98, and F1-score of 0.93. It also correctly identifies all fractures in lateral-view images, which is a view type that posed notable challenges for the human participants, while producing results in under one second per image.

ÖZET

DISTAL RADIUS KIRIK TESPITI: BILGISAYAR GÖRÜSÜ ALGORITMALARININ KARŞILAŞTIRILMASI

Distal radius kırıkları, acil servis ve ortopedi kliniklerinde en sık karşılaşılan iskelet yaralanmaları arasında yer almakta olup, tüm yaş gruplarında el bileği kaynaklı travmaların önemli bir bölümünü oluşturmaktadır. Yaygın görülmelerine rağmen, özellikle deneyimsiz klinisyenler tarafından, anatomik varyasyonlar ve radyografik karmaşıklık nedeniyle ince kırıklar sıklıkla gözden kaçırılmaktadır. Erken ve doğru tespit, uygun tedavinin sağlanması ve uzun vadeli fonksiyonel kayıpların önlenmesi açısından kritik öneme sahiptir.

Önerilen çalışma, distal radius kırıklarının el bileği radyografileri üzerinde otomatik olarak tespit edilmesi ve lokalize edilmesi amacıyla geliştirilen derin öğrenme tabanlı bir sistemi sunmaktadır. Klinik iş akışlarını desteklemenin yanı sıra tıp eğitimi bağlamında da fayda sağlaması hedeflenen bu sistem kapsamında, Faster R-CNN, YOLOv8, YOLOv11, Mask R-CNN ve EfficientNet gibi modeller değerlendirilmiştir. Her bir model etiketlenmiş bir veri kümesi üzerinde eğitilmiş ve doğrulanmış; hassasiyet, duyarlılık ve F1-skoru gibi standart metrikler üzerinden karşılaştırılmıştır. Bütün modellerin karşılaştırılmış başarısı açısından en iyi performansı EfficientNet göstermiş ve bu model, özel olarak geliştirilen bir Python uygulamasına entegre edilmiştir. Uygulama, kullanıcıların radyografileri yüklemesine, segmentasyon tabanlı tahminleri görüntülemesine ve tanısal geri bildirim göndermesine olanak tanıyarak, hem klinik hem de eğitim amaçlı kullanıma uygun bir yapı sunmaktadır.

Sistemin tanısal potansiyelini değerlendirmek amacıyla, ortopedi uzmanları tarafından işaretlenen 40 ayrı radyografiden oluşan bir test veri kümesi kullanılarak ek bir değerlendirme gerçekleştirilmiştir. EfficientNet modeli, aynı görüntüleri manuel olarak değerlendiren beş adet altıncı sınıf tıp fakültesi öğrencisi ile karşılaştırılmıştır. Model, tüm temel metriklerde internleri geride bırakmış; %89 hassasiyet, %98 duyarlılık ve %93 F1-skoru elde etmiştir. Özellikle, insan katılımcıların zorlandığı lateral açıya ait radyografilerdeki tüm kırıkları doğru şekilde tespit edebilmiş ve her bir görüntü için bir saniyenin altında sonuç üretmiştir.

DEDICATION

I dedicate this work to the memory of my aunt, who passed away during writing this thesis. She always believed in me and hoped to see this day come. Her love and encouragement continue to live on in my heart, and this thesis is, in many ways, dedicated to her.

Burcu Selcuk

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor, Assistant Professor Doctor Tacha Serif, for his continuous guidance, support, patience and encouragement throughout this study. His mentorship has been essential in shaping both the direction of my research and my academic development.

I would also like to extend my heartfelt thanks to Dr. Safa Serif, whose medical expertise and insight as an orthopedic surgeon greatly enriched the clinical relevance of this work. His contributions provided an essential foundation for the validation and evaluation of the proposed system.

Last but not least, I am deeply thankful to my family, whose love, patience, and unwavering belief in me have carried me through every challenge in this study. Their presence has been a constant source of strength in moments of both light and darkness. I am endlessly grateful to them, beyond words, beyond measure.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY iii
ABSTRACT iv
ÖZET v
DEDICATION vi
ACKNOWLEDGEMENTS vii
TABLE OF CONTENTS viii
LIST OF FIGURES x
LIST OF TABLES xviii
LIST OF ABBREVIATIONS xx
1. INTRODUCTION 1
2. BACKGROUND 1
2.1. DISTAL RADIUS FRACTURES: INFLUENCES AND CHALLENGES 2
2.2. DIAGNOSTIC PROCESS OF DRFS 4
2.3. CLASSIFICATION OF DRFS 6
2.4. TREATMENT OF DRFS 10
2.5. FRACTURE DETECTION WITHOUT ARTIFICIAL INTELLIGENCE 16
2.6. AI-ASSISTED DISTAL RADIUS FRACTURE STUDIES 20
3. METHODOLOGY 34
3.1. IMAGING MODALITIES AND DATASETS 34
3.2. IMAGE PROCESSING AND ANNOTATION 38
3.3. MODEL ARCHITECTURES 49
3.4. MODEL EVALUATION METRICS 72
4. ANALYSIS AND DESIGN 81
4.1. SYSTEM REQUIREMENTS 81
4.2. SYSTEM WORKFLOW AND RESEARCH PIPELINE 84
4.3. CLASS DIAGRAM FOR THE PYTHON APPLICATION 89
4.4. SEQUENCE DIAGRAM FOR THE PYTHON APPLICATION 90
5. DATASET PREPERATION 92
6. IMPLEMENTATION OF DETECTION MODELS 95
6.1. TRAINING OF FASTER R-CNN MODEL 96

6.2. TRAINING OF YOLO ARCHITECTURE BASED MODELS 100
7. IMPLEMENTATION OF SEGMENTATION MODELS 105
7.1. TRAINING OF MASK R-CNN MODEL 105
7.2. TRAINING OF EFFICIENTNET MODEL 108
7.3. TRAINING OF YOLO ARCHITECTURE BASED SEGMENTATION MODELS 113
8. APPLICATION IMPLEMENTATION 122
9. EVALUATION AND TESTING OF THE DETECTION MODELS 126
9.1. FASTER R-CNN PERFORMANCE EVALUATION 127
9.2. FASTER R-CNN REAL-WORLD TESTING 133
9.3. YOLO DETECTION PERFORMANCE EVALUATION 138
9.3.1. YOLOv8 Detection Evaluation 138
9.3.2. YOLOv11 Detection Evaluation 146
9.4. YOLO DETECTION REAL-WORLD TESTING 152
9.5. COMPARISON AND ASSESSMENT OF DETECTION MODELS 163
10. EVALUATION AND TESTING OF THE SEGMENTATION MODELS 165
10.1. MASK R-CNN PERFORMANCE EVALUATION 166
10.2. MASK R-CNN REAL-WORLD TESTING 169
10.3. EFFICIENTNET PERFORMANCE EVALUATION 174
10.4. EFFICIENTNET REAL-WORLD TESTING 178
10.5. YOLO SEGMENTATION PERFORMANCE EVALUATION 180
10.6. YOLO SEGMENTATION REAL-WORLD TESTING 190
10.7. COMPARATIVE CONCLUSION OF SEGMENTATION MODELS 197
11. COMPARISON OF MODEL AND MEDICAL INTERN PERFORMANCE 200
12. TECHNICAL DISCUSSION: INTERPRETABILITY VS EXPLAINABILITY ... 210
13. CONCLUSIONS 212
REFERENCES 216

LIST OF FIGURES

Figure 2.1. Age-specific incidence rates of DRFs. 4

Figure 2.2. CBCT versus CT. 5

Figure 2.3. Nissen-Lie classification schematic drawing. 7

Figure 2.4. Gartland and Werley classification schematic drawing. 7

Figure 2.5. Lidström classification schematic drawing. 8

Figure 2.6. Older classification schematic drawing. 9

Figure 2.7. Fykman classification schematic drawing. 9

Figure 2.8. Incidence distribution of DRF across age. 14

Figure 2.9. Incidence distribution of DRF across gender and age. 15

Figure 2.10. Yearly trends of treatment for DRFs. 15

Figure 2.11. Average time and accuracy of reading a radiograph. 22

Figure 2.12. Heatmaps generated by the system versus radiologists. 22

Figure 2.13. Diagnostic process of ensemble model. 23

Figure 2.14. Diagnostic process of the CNN model. 27

Figure 2.15. Grad-CAM mapping on a sample radiograph. 28

Figure 2.16. EfficientNet and YOLOv5 ensemble model process. 29

Figure 3.1. Sample X-ray image. 35

Figure 3.2. Anteroposterior and posteranterior views. Anteroposterior angle, Posteroanterior view. 36

Figure 3.3. Lateral view X-ray imaging. 36

Figure 3.4. Oblique view X-ray imaging. 37

Figure 3.5. (a) Schaphoid angle, (b) Radial deviation angle 38

Figure 3.6. Bounding box annotation of a fracture. 39

Figure 3.7. Annotation of a bounding box. 40

Figure 3.8. Polygon annotations on fractures. 42

Figure 3.9. Gaussian operation on the same image. 46

Figure 3.10. RGB versus HSV color space. 47

Figure 3.11. (a) Original image, (b) Image after rotation and flipping augmentations. 48

Figure 3.12. Image after brightness and contrast augmentations. 48

Figure 3.13. CNN architecture structure. 49

Figure 3.14. Convolution operations using 3x3 kernel. 50

Figure 3.15. Convolution operation on a zero padded input tensor. 51

Figure 3.16. Common activation function graphs. 52

Figure 3.17. (a) Max pooling, (b) average pooling. 53

Figure 3.18. GD optimization algorithm process. 55

Figure 3.19. Training error comparison of a shallow versus deep network. 57

Figure 3.20. Skip connections in a 18-layer ResNet. 58

Figure 3.21. Illustration for a bottleneck block. 59

Figure 3.22. CNN scaling. 59

Figure 3.23. Architecture of U-Net. 61

Figure 3.24. A sample Fast R-CNN pipeline. 63

Figure 3.25. The region proposal procedure in Faster R-CNN. 64

Figure 3.26. (a) Fast R-CNN prediction, (b) Mask R-CNN prediction. 65

Figure 3.27. Mask R-CNN achitecture for instance segmentation. 65

Figure 3.28. YOLO architecture. 67

Figure 3.29. YOLO network structure. 68

Figure 3.30. (a) YOLOv2 mean IoU, (b) box dimension clustering. 70

Figure 3.31. YOLOv8 architecture 71

Figure 3.32. YOLOv11 architecture 72

Figure 3.33. Confusion matrix. 73

Figure 3.34. A sample AUC-ROC curve for a fictional training. 76

Figure 3.35. A sample PR curve for a fictional training. 76

Figure 3.36. Illustration of IoU. 77

Figure 4.1. The general workflow of the proposed study. 85

Figure 4.2. The detailed methodology flow of the proposed study. 87

Figure 4.3. The Gantt chart for the study. 88

Figure 4.4. The flowchart for the Python-based application. 89

Figure 4.5. The class diagram for the Python-based application. 90

Figure 4.6. The sequence diagram for the Python-based application. 91

Figure 5.1. (a) Bounding box annotations for detection models, (b) bounding box annotations converted to polygon annotations for segmentation models. 92

Figure 6.1. (a) Time taken for data loading per iteration, (b) data loading time during training, (c) the estimated time remaining for training. 99

Figure 6.2. (a) Total loss, (b) bounding box regression loss, (c) RPN localization loss for Faster R-CNN training. 99

Figure 6.3. Number of positive and negative anchors by the RPN. 100

Figure 6.4. Number of background and foreground sample by the RoI. 101

Figure 6.5. Bounding box distribution of the dataset, produced by the trained YOLOv8 models. 103

Figure 6.6. Bounding box attributes of the dataset, produced by the trained YOLOv8 models..... 103

Figure 7.1. The number of positive and negative anchors proposed by the ROI. 107

Figure 7.2. (a) Mask R-CNN false negatives, (b) false positives during training. 108

Figure 7.3. The total loss graph of Mask R-CNN model training. 109

Figure 7.4. A prepared image from the dataset for EfficientNet training. 109

Figure 7.5. EfficientNet loss values during the training..... 112

Figure 7.6. (a) YOLOv8m training segmentation loss, (b) training DFL loss. 114

Figure 7.7. (a) YOLOv8m validation segmentation loss, (b) validation DFL loss..... 115

Figure 7.8. (a) YOLOv8l training segmentation loss, (b) training DFL loss. 116

Figure 7.9. (a) YOLOv8l validation segmentation loss, (b) validation DFL loss. 116

Figure 7.10. (a) YOLOv11m training segmentation loss, (b) training DFL loss. 118

Figure 7.11. (a) YOLOv11m validation segmentation loss, (b) validation DFL loss. 119

Figure 7.12. (a) YOLOv11l training segmentation loss, (b) training DFL loss. 119

Figure 7.13. (a) YOLOv11l validation segmentation loss, (b) validation DFL loss..... 120

Figure 8.1. The Python application layout. 123

Figure 9.1. Flowchart of the procedure for evaluating and testing the detection models..... 126

Figure 9.2. Faster R-CNN bounding box precision at different IoU thresholds. 128

Figure 9.3. Faster R-CNN bounding box precision at IoU ≥ 0.5. 128

Figure 9.4. Faster R-CNN bounding box precision at IoU ≥ 0.75..... 129

Figure 9.5. (a) Faster R-CNN AP on small, (b) medium, and, (c) large fractures. 130

Figure 9.6. Key evaluation metrics graph for Faster R-CNN evaluation..... 131

Figure 9.7. Faster R-CNN inference speed per iteration on the validation set. 132

Figure 9.8. Faster R-CNN confusion matrix on the test set..... 133

Figure 9.9. False positive due to overlapping bounding boxes. 135

Figure 9.10. False positive due to high sensitivity to normal bone structure. 136

Figure 9.11. False negative due to casts and low contrast..... 137

Figure 9.12. False negative due to surgical hardware..... 137

Figure 9.13. YOLOv8m model recall-confidence graph on the validation set. 139

Figure 9.14. YOLOv8m model precision-recall curve on the validation set..... 139

Figure 9.15. YOLOv8m model precision-confidence graph on the validation set. 140

Figure 9.16. YOLOv8m F1-Score across confidence thresholds on the validation set. .. 141

Figure 9.17. YOLOv8m model confusion matrix on the validation set..... 142

Figure 9.18. YOLOv8m results during training and on the validation set. 142

Figure 9.19. (a) YOLOv8l precision-confidence, (b) precision-recall curves on validation set..... 144

Figure 9.20. (a) YOLOv8l F1-Score-confidence, (b) recall-confidence curves on validation set..... 144

Figure 9.21. YOLOv8l normalized confusion matrix. 145

Figure 9.22. (a) YOLOv11m precision-confidence, (b) precision-recall curves on validation set..... 147

Figure 9.23. (a) YOLOv11m recall-confidence, (b) F1-Score-confidence curves on validation set. 147

Figure 9.24. YOLOv11m normalized confusion matrix. 148

Figure 9.25. (a) YOLOv11l precision-confidence, (b) precision-recall curves on validation set. 149

Figure 9.26. (a) YOLOv11m recall-confidence, (b) F1-Score-confidence curves on validation set. 150

Figure 9.27. YOLOv11l normalized confusion matrix. 151

Figure 9.28. (a) YOLOv8m cast and (b) surgical hardware TP predictions. 154

Figure 9.29. YOLOv8m raw confusion matrix on the test set. 155

Figure 9.30. YOLOv8l raw confusion matrix on the test set. 157

Figure 9.31. YOLOv11m prediction of multiple bounding boxes. 159

Figure 9.32. (a) Prediction of YOLOv8 models (b) versus the YOLOv11m model on the same image. 160

Figure 9.33. YOLOv11m prediction on the casted, low constrast test instance. 160

Figure 9.34. YOLOv11m raw confusion matrix on the test set. 161

Figure 10.1. Flowchart of the procedure for evaluating and testing the segmentation models. 165

Figure 10.2. Mask R-CNN mask loss curve during training. 166

Figure 10.3. Mask R-CNN mask accuracy curve during training. 167

Figure 10.4. Mask R-CNN FN rate curve during training. 167

Figure 10.5. Mask R-CNN FP rate curve during training. 168

Figure 10.6. (a) Mask R-CNN AP at IoU 0.50:0.95, (b) AP at IoU threshold set to 50… 168

Figure 10.7. Mask R-CNN confusion matrix on the validation set. 169

Figure 10.8. Mask R-CNN confusion matrix on the test set. 170

Figure 10.9. Mask R-CNN model prediction on the test set with an FP. 171

Figure 10.10.(a) Mask R-CNN's correct segmentation on hand, (a) correct segmentation on ulna. 171

Figure 10.11.(a) Mask R-CNN FP, (b) other FP on instances with casts. 172

Figure 10.12.(a) Mask R-CNN's missed segmentation, (b) other missed segmentation on instances with surgical hardware. 173

Figure 10.13.EfficientNet inference times on the validation set on the validation set. 175

Figure 10.14.EfficientNet precision-recall curve on the validation set. 175

Figure 10.15.EfficientNet segmentation performance on the validation set. 176

Figure 10.16.EfficientNet FP prediction instance. 179

Figure 10.17.EfficientNet another FP prediction instance. 180

Figure 10.18.YOLOv8m normalized confusion matrix on the validation set. 181

Figure 10.19.(a) YOLOv8m segmentation model precision-confidence, (b) recall-confidence curves. 182

Figure 10.20.YOLOv8m segmentation precision-recall curve. 182

Figure 10.21.YOLOv8m segmentation F1-confidence curve. 183

Figure 10.22.YOLOv8l segmentation normalized confusion matrix on the validation set. 184

Figure 10.23.YOLOv8l segmentation F1-confidence curve. 185

Figure 10.24.(a) YOLOv11m segmentation, (b) YOLOv11l segmentation confusion matrix. 186

Figure 10.25.(a) YOLOv11m segmentation, (b) YOLOv11l segmentation precision-confidence curve. 187

Figure 10.26.(a) YOLOv11m segmentation, (b) YOLOv11l segmentation recall-confidence curve. 188

Figure 10.27.(a) YOLOv11m segmentation, (b) YOLOv11l segmentation precision-recall curve. 189

Figure 10.28.(a) YOLOv11m segmentation, (b) YOLOv11l segmentation F1-confidence curve. 189

Figure 10.29.(a) YOLOv8m segmentation, (b) YOLOv8l segmentation confidence matrices on test set. 191

Figure 10.30.(a) YOLOv8m segmentation, (b) YOLOv8l segmentation on the same image. 192

Figure 10.31.YOLOv8 segmentation on bracelet radiograph instance. 193

Figure 10.32.YOLOv8m segmentation on a casted image. 194

Figure 10.33.YOLOv11m and YOLOv11l model confusion matrices on the test set. 195

Figure 10.34.The same segmentation over the correct one. 196

Figure 10.35.A unique false positive observed in YOLOv11m segmentation. 196

Figure 11.1. A bar chart comparing the completion times. 202

Figure 11.2. A overlap score of 0.4 on a sample radiograph. 202

Figure 11.3. (a) Bar chart, (b) pie chart of collective intern performance. 206

Figure 11.4. Bar chart comparing the performance of the model across interns. 208

Figure 12.1. (a) Model's direct prediction, (b) Grad-CAM mapping of the prediction... 211

LIST OF TABLES

Table 3.1. Parameters and accuracy of EfficientNet models.....61

Table 3.2. Comparison of R-CNN, Fast R-CNN, and Faster R-CNN.....64

Table 5.1. Image augmentation techniques 93

Table 6.1. Hyperparameters for YOLO detection models 104

Table 7.1. Mask R-CNN model hyperparameter adjustment..... 106

Table 7.2. YOLOv8 segmentation hyperparameters 113

Table 9.1. Faster R-CNN performance metrics on test set 134

Table 9.2. Validation metrics for YOLOv8m..... 143

Table 9.3. Validation metrics for YOLOv8l 146

Table 9.4. Validation metrics for YOLOv11m 149

Table 9.5. Validation metrics for YOLOv11l..... 151

Table 9.6. Performance metrics for YOLOv8m on the test set..... 156

Table 9.7. YOLOv8m performance metrics on test set 158

Table 9.8. YOLOv11m performance metrics on test set..... 161

Table 9.9. YOLOv11l performance metrics on test set 162

Table 9.10. Comparison of YOLO detection models on the test set 163

Table 9.11. Comparison of inference times of YOLO detection models 164

Table 10.1. Testing metrics for Mask R-CNN model 173

Table 10.2. Performance metrics for EfficientNet model 177

Table 10.3. Performance metrics for EfficientNet on test set 180

Table 10.4. YOLOv8m versus YOLOv8l segmentation performance comparison on validation set 186

Table 10.5. YOLOv11m versus YOLOv11l segmentation performance comparison on validation set 190

Table 10.6. YOLOv8m and YOLOv8l segmentation models performances on the test set 194

Table 10.7. YOLOv11m and YOLOv11l segmentation models performances on the test set 197

Table 10.8. Comparison of segmentation models based on various performance metrics 197

Table 11.1. Intern detection summary 203

Table 11.2. Cumulative intern detection performance 206

Table 11.3. Interns versus model performance 207

Table 11.4. Comparison metrics of interns versus EfficientNet model 208

LIST OF ABBREVIATIONS

AP	Average Precision
AP@50	Average Precision at IoU threshold 0.5
AP@50:95	Average Precision averaged over IoU thresholds from 0.5 to 0.95
APs, APm, APl	AP for small, medium, and large object sizes respectively
AR	Average Recall
ASN	Average Sample Number
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
CBCT	Cone Beam Computed Tomography
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CV	Computer Vision
DL	Deep Learning
DRF	Distal Radius Fracture
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GPU	Graphics Processing Unit
IoU	Intersection over Union
mAP	Mean Average Precision
NMS	Non-Maximum Suppression
PR	Precision-Recall
RAM	Random Access Memory
ResNet	Residual Networks
ROC	Receiver Operating Characteristic
TN	True Negative
TP	True Positive
YOLO	You Only Look Once

1. INTRODUCTION

Distal radius fractures (DRFs) are among the most common fractures encountered in clinical practice, particularly in emergency departments. Due to their frequency and the high volume of radiographs processed daily, timely and accurate identification of these fractures is essential for proper treatment planning and patient outcomes. However, manual interpretation of wrist radiographs is a demanding task, often performed under time constraints and with variable levels of clinical experience, especially in emergency care settings. This increases the risk of misdiagnosis or delayed diagnosis, which can lead to complications such as malunion, stiffness, or chronic pain.

Recent advances in computer vision (CV) and deep learning (DL) have shown great promise in the automatic analysis of medical images, particularly in the detection of bone fractures. Object detection and segmentation algorithms have the potential to support radiologists by highlighting fracture regions, thereby improving diagnostic efficiency and reducing cognitive workload. Despite the increasing popularity of deep learning-based solutions in medical imaging, there is still a lack of standardization and direct comparison among different model architectures in the context of DRF detection. Moreover, many existing studies rely on older architectures, overlooking the performance potential of more recent, state-of-the-art models.

The proposed study addresses these gaps by investigating whether modern computer vision models can accurately detect and segment distal radius fractures in radiographic images, and how their diagnostic performance compares with that of medical interns. The overarching aim is to evaluate the feasibility of integrating such models into real-world clinical and educational workflows, where they could serve as assistive tools to support training and reduce the burden on radiologists.

The challenges of diagnostic overload and variability in fracture interpretation have prompted increased interest in AI-driven decision support tools. In emergency departments, where the rapid triage and evaluation of wrist injuries is a routine task, there is a potential requirement for systems that can offer reliable second opinions or flag suspicious findings. Models that are able to detect fractures with high sensitivity, especially those that minimize false negatives, could substantially reduce diagnostic errors and improve patient outcomes. Their potential

utility extends not only to practicing clinicians but also to medical trainees who are still developing radiograph interpretation skills.

While numerous studies have explored fracture detection using deep learning, a gap persists in the application of contemporary, high-performing model architectures. Approaches such as YOLOv8, YOLOv11, and segmentation-focused models like EfficientNet have demonstrated notable success in general computer vision tasks but remain underutilized in the DRF detection domain. By incorporating these models into a comparative framework alongside more established approaches like Faster R-CNN and Mask R-CNN, this study aims to address a pressing need in the literature and evaluate whether these newer architectures provide measurable advantages in medical imaging contexts.

Another important consideration lies in the potential for these tools to serve not just as diagnostic aids but also as educational instruments. Comparing AI-generated results to those produced by medical interns offers insight into how these systems could be integrated into the learning process, helping trainees develop pattern recognition skills while reinforcing anatomical understanding.

The proposed study undertakes a systematic investigation of fracture detection and segmentation in wrist radiographs using a variety of DL models. A curated dataset of distal radius fracture images, annotated by orthopedic surgeons, provides the foundation for training and evaluation. Multiple models, including Faster R-CNN, YOLOv8, YOLOv11, Mask R-CNN, and EfficientNet, are trained and evaluated on this dataset using standard performance metrics such as mean average precision (mAP), F1-score, intersection over union (IoU), and recall.

To assess generalizability, the models are further tested on a separate, real-world dataset annotated by a senior orthopedic surgeon. This dataset includes particularly challenging cases such as casted wrists, surgical hardware, and two-view radiographs. The best-performing model is then selected for comparative analysis with five medical interns, using the surgeon-annotated set as the reference standard. Inference accuracy, speed, and prediction errors are compared across both the model and human observers.

As part of this study, a lightweight Python application is developed to operationalize the

selected model. This application enables users to upload radiographs, view segmentation results, and review confidence scores and inference times. A feedback mechanism allows medical professionals to submit comments before or after viewing the predictions. All predictions and feedback are securely logged in encrypted files, offering a traceable audit trail that aligns with principles of transparency and data privacy.

Accordingly, the proposed study is structured as follows. While the next chapter presents the background and related work in fracture detection, deep learning in medical imaging, and the clinical relevance of DRFs, Chapter 3 describes the datasets, annotation processes, evaluation criteria and preprocessing pipeline in the literature and within this study. The following chapter details the functional requirements, system components, and architecture diagrams. Thereafter, in Chapter 5, the process for dataset preparation is discussed in detail. Chapters 6 and 7 detail the technical implementation of the models, including model training configurations and development environment. In Chapters 8 and 9, quantitive and qualitative evaluations of the models are presented. In Chapter 10, the best performing model is compared to medical interns on an additional test data. Finally, Chapter 11 draws the final conclusions, summarizes key findings and suggests directions for future research and development.

2. BACKGROUND

In anatomical terminology, distal refers to farthest from the point of origin, or further from the center of the body. Thus, it is the opposite of proximal, which means closer to the center of the body. While the distal radius is the part of the radius that is close to the wrist, the proximal radius refers to the part of the radius near the elbow [1].

Hence, a distal radius fracture is a break in the radius bone near the wrist joint, typically occurring within the distal 3 cm of the radius or radiocarpal joint. The radius is one of the two major bones of the forearm, and the distal end connects to the wrist joint, playing a critical role in hand and wrist movement [2].

Distal radius fractures can be extra-articular, intra-articular, and stable or unstable [3]. While extra-articular fractures do not involve the wrist joint surface, intra-articular fractures include the wrist joint, which makes them more complex. Moreover, depending on the alignment and potential for displacement, fractures are attributed to be stable or unstable.

The most frequent cause of distal radius fractures is a fall onto an outstretched hand (FOOSH), which is a mechanism of injury that transmits force through the wrist and fractures the radius near the joint [4]. This type of injury is particularly common among older adults with osteoporosis and among younger individuals following high-energy trauma, such as sports injuries or road traffic accidents.

The diagnosis, treatment, and management of distal radius fractures (DRFs) have been an ongoing focus of medical research, reflecting their prevalence and significance in clinical practice. Over time, a significant number of researchers have contributed to the understanding of these fractures.

To the best of our knowledge, the first documented mention of DRFs in medical literature is by Colles [5] in 1814. In Colles's study, the unique attributes of this fracture type, later named the "Colles' fracture" in his honor, are detailed. Colles accentuates the injury's specific characteristics, such as the absence of crepitus, which is the lack of clicking, popping, or crackling sound in a joint. Furthermore, he also discusses the deformation which occurs approximately 3.8 centimeters above the carpal extremity. The carpal bones are wrist bones

that attach the distal ends of the radial and ulnar bones of the forearm to the bases of five metacarpal bones in hand [6]. Additionally, he clarifies the diagnostic challenges posed by these fractures. Colles develops a diagnostic technique by extending and manipulating the patient's hand to detect the nuanced mobility of the fractured bone. This technique lays the groundwork for future diagnostic improvements.

2.1. DISTAL RADIUS FRACTURES: INFLUENCES AND CHALLENGES

DRFs are among the most common upper extremity fractures in clinical practice [7], and their influence on patients' quality of life and healthcare systems is substantial. Age-related changes, lifestyle factors, socioeconomic elements, and even environmental circumstances affect their prevalence, intensifying the diverse nature of their causes.

Clement et al. [8] conducts a study to discover the factors influencing DRFs from a socioeconomic point of view. Consequently, they find that fractures are induced by diverse factors, including demographics, mechanism of injury, and bone quality. Focusing on the fact that socioeconomic status has been shown to play a significant role in shaping the outcomes of DRFs; they reveal that individuals from socioeconomically constrained backgrounds are more likely to sustain DRFs at younger ages, often due to high-energy mechanisms of injury, and show a higher prevalence among males. In contrast, prosperous patients are more plausible to have complications such as complex regional pain syndrome which is a persistent pain that arises after a medical event.

However, the consequences of DRFs do not differ across socioeconomically different groups. Some of these outcomes are listed to be the range of motion and grip strength. Nevertheless, there are some subtle trends of change between the two groups, the most observable one being the diminished wrist flexion in more underprivileged patients. The study by Clement et al. holds value for practicing targeted prevention strategies and personalized care interventions.

The study by Turner et al. [9] highlights both the influences and the challenges regarding the DRFs. The study reveals that complication rates range from 6% to 80% and describes that they can emerge both from the fractures themselves and their treatment. Their work categorizes complications into three groups -namely immediate, early, and late stages. The

immediate stage consists of nerve injuries like acute carpal tunnel syndrome while the early stage of complications comprises cast-related issues or loss of reduction. Last but not least, late-stage complications involve arthritis, tendon ruptures, and complex regional pain syndrome.

Turner et al. proposed a standardized checklist to systematically assess and report complications, aiming to improve clinical outcomes for DRF patients. The study highlights the importance of premature and accurate diagnosis to lessen complications while recommending careful monitoring for distinctive signs and proper treatment planning to prevent long-term dysfunction.

From a different perspective, Flinkkilä et al. [10] explore the effects of seasonal variation on DRFs in Finland. The study reviews 285 patients to find the impact of weather conditions on DRFs. Additionally, all patients are at least 16 years old. Using a unique weather and pavement condition model developed by the Finnish Meteorological Institute, the researchers investigate how slippery pavement circumstances influence the incidence of fractures.

They have used person-years measurement, which is a measurement used in epidemiology to account for both the number of people in a study and the amount of time they were at risk. It combines these two elements to regularize incidence rates and make comparisons between groups or studies more exact [11]. Consequently, they discovered that the natural incidence rate is 258 per 100,000 person-years, with a higher rate in females (363/100,000) compared to males (147/100,000). Elderly women exhibited the highest fracture rates, peaking at 1,107/100,000 person-years for women aged 80 and above.

When it comes to seasonal variation, they identify that DRF rates increase greatly during winter, especially on days with slippery pavement. The fracture risk was 2.5 times higher on slippery days compared to non-winter days, and 1.4 times higher on normal winter days. The day following a slippery day also showed a marked increase in fracture rates, which is attributed to delayed medical consultation or additional falls.

In the same study, the mechanism of injury is also examined in detail and the findings show that the majority of fractures (77%) result from low-energy falls, such as falls from standing height, particularly in icy conditions.

Aside from the weather, other contributing factors such as activity levels and footwear design are also identified. Women, especially elderly women, are excessively affected, likely due to age-related bone density loss and outdoor activities.

2.2. DIAGNOSTIC PROCESS OF DRFS

Brogren et al. [12] discuss the epidemiology of DRFs and include details on the radiographic evaluation of the fractures. In the study, they discuss that radiography remains the gold standard for the initial diagnosis of DRFs. Standard posteroanterior, lateral, and oblique views are commonly employed to identify fracture patterns and assess displacement. While radiographs provide valuable information about the fracture's location, angulation, and step-off, they assert that there are limitations when it comes to detecting subtle intra-articular fractures or complex multi-fragmentary patterns.

Their population-based study records 335 cases of acute DRFs over a year and employed the Müller Arbeitsgemeinschaft für Osteosynthesefragen (AO) classification system [13], which is a standard framework for categorizing fractures based on their anatomical and structural characteristics in order to assess the severity of fractures and guide treatment accordingly. The research emphasizes measurements such as volar tilt and ulnar variance, which are of utmost importance for deciding fracture displacement. These parameters impact treatment decisions as they lead to the distinguishment between surgical and non-operative interventions. Their findings show that almost two-thirds of fractures are displacements. Figure 2.1 shows the incidence age-specific incidence rates of DRFs. The study highlights that incidence rate increases significantly in older populations, especially for those who are above age 50 and women.

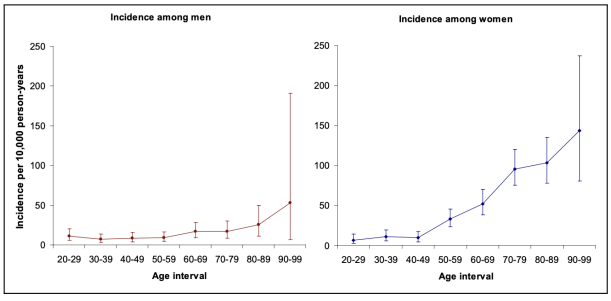


Figure 2.1. Age-specific incidence rates of DRFs [12].

Cone-beam computed tomography (CBCT) is a specialized type of computed tomography (CT) imaging that uses a cone-shaped X-ray beam and a flat-panel detector to acquire high-resolution, three-dimensional images in a single rotational scan [14]. Traditional CT uses a fan-shaped beam to create separate slices. On the other hand, CBCT is able to capture a full volume of data in one rotation. This results in a reduction of scan time while compensating for artifacts related to the movement of the patient. Consequently, the patient is also exposed to less radiation. Figure 2.2 depicts the difference between the outcomes of a CT and CBCT scan.

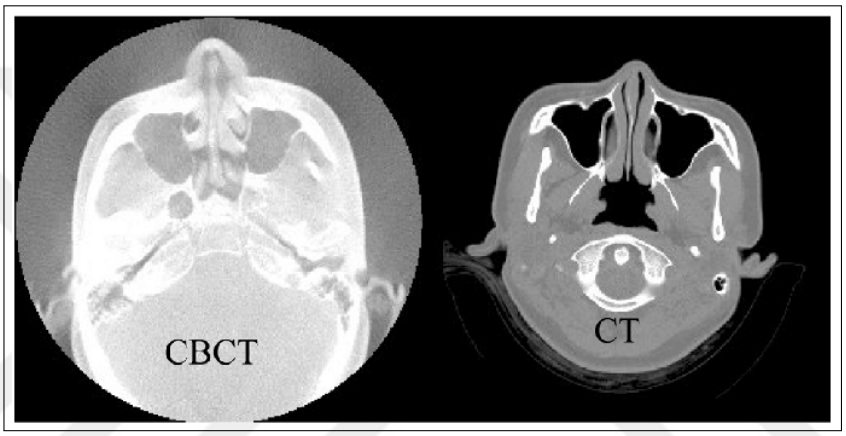


Figure 2.2. CBCT vs CT [15].

Thus, a study conducted by Grunz et al. [16] compares the diagnostic accuracy of CBCT and standard radiography in 113 patients with DRFs and scaphoid fractures. For basic fracture detection, radiography shows equivalent accuracy. On the other hand, for the identification of complex fracture patterns, CBCT exhibits superior performance. Furthermore, the study also highlights CBCT's capability of delivering high-quality imaging with less radiation exposure.

Point-of-care ultrasound (POCUS) is an ultrasound imaging procedure conducted at the patient's bedside by the doctor [17]. Since this technique provides instantaneous results, it can enable real-time diagnostics. Furthermore, it eliminates the radiation variable from the equation, which can be beneficial for patients with risks associated with radiation exposure, such as pediatric patients.

From this perspective, a study by Kozaci et al. [18] evaluates the effectiveness of POCUS compared to standard radiography in diagnosing DRFs among 83 patients. The results of the study reveal that POCUS achieved high sensitivity (98%) and specificity (96%) in detecting

DRFs. Moreover, the identification of fracture types is improved with POCUS, which leads to easier decision-making for interventions.

However, they state that while POCUS offers exceptional advantages where radiation exposure must be minimized due to various reasons and its ability to dynamically assess soft tissues, cartilage, and fracture alignment in real-time provides diagnostic value, particularly for complex fractures; its significance relies on the clinician's expertise and skill level.

2.3. CLASSIFICATION OF DRFS

Over the years, multiple systems for the classification of DRFs have been developed. Each system has a distinctive emphasis on the characteristics of the fracture, whether it be joint involvement, comminution, displacement etcetera. Systematic classification of DRFs lets clinicians predict treatment outcomes, identify the probability of complications as well as determine the intervention most suitable for a fracture.

The Nissen-Lie classification [19] [20] is one of the first methods developed for the categorization of DRFs. This classification divides fractures into 6 types based on displacement.

Type 1 fracture only ensues in children between the ages of 1 and 15 and it is described to be a fracture at the intersection of the shaft and distal extremity of the radius. Type 2 fracture, on the other hand, only occurs in the age range of 10 to 20 and it is the slipping of the epiphysis, which is a long bone in the arm. Type 3 fracture involves minimal displacement and is characterized by almost undamaged alignment and no considerable disturbance to the bone's anatomical structure. Type 4 fracture consists of extra-articular fractures with dorsal angulation, in which the fracture does not extend into the joint surface. Type 5 fracture comprises intra-articular fractures with comminution, which indicates that the fracture does extend into the joint and there is fragmentation. Lastly, Type 6 fractures are fractures of the radial styloid which is the lateral surface of the distal radius bone. They often result from direct impact on the radial aspect of the wrist. Figure 2.3 depicts the schematic drawings for the Type 3, Type 4, Type 5, and Type 6 fractures in Nissen-Lie classification system.

Gartland and Werley [21] classification system is another technique proposed in 1951 which

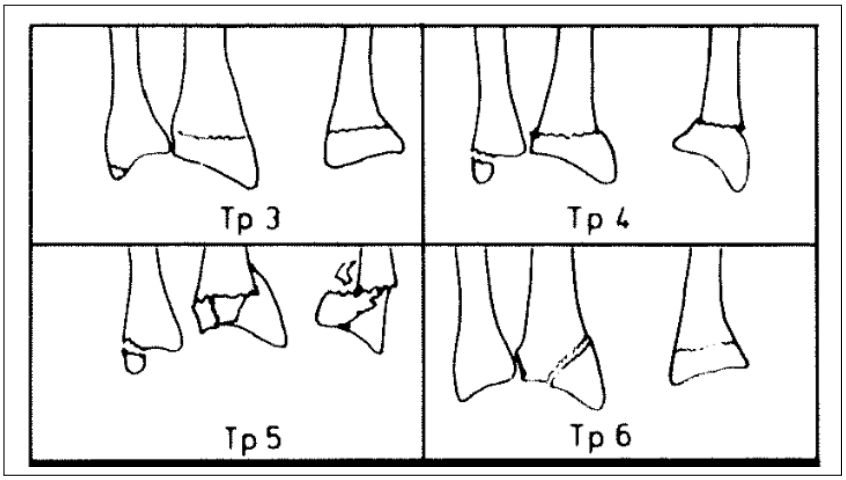


Figure 2.3. Nissen-Lie classification schematic drawing [20].

categorizes fractures into 3 groups based on displacement and joint involvement. Group 1 includes displaced fractures that are extra-articular, where extra-articular refers to fractures that occur outside of a joint. In contrast, Group 2 contains non-displaced fractures that are intra-articular, where intra-articular refers to fractures that extend into and involve the articular surface of a joint. Finally, Group 3 consists of displaced fractures that are intra-articular. Figure 2.4 depicts the schematic drawings for the groups in the Gartland and Werley classification system.

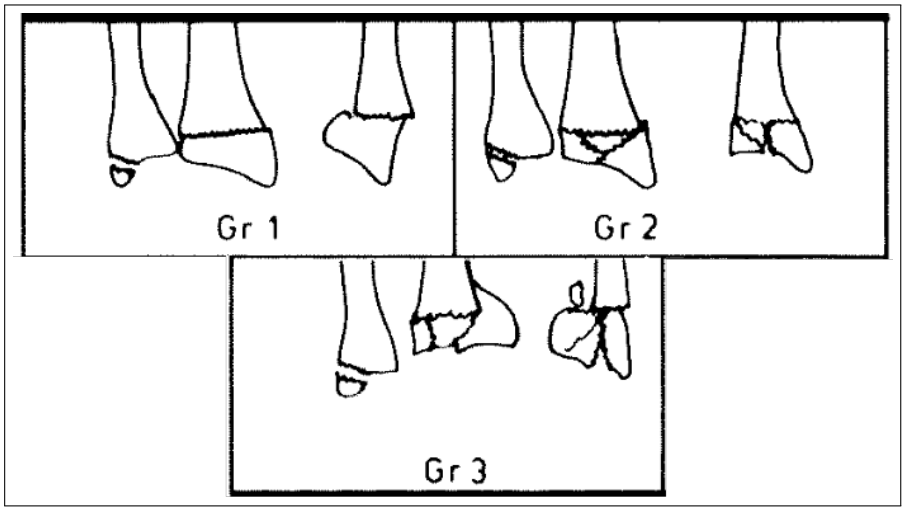


Figure 2.4. Gartland and Werley classification schematic drawing [20].

Lidström [22] classification organizes fractures into six classes and two immediate types based on displacement and comminution characteristics of the fracture. Type 1 fractures involve minimal displacement. Type 2 fractures branch into 5 categories where the type

2-A group involves extra-articular fractures with dorsal angulation. Type 2-B fractures are intra-articular fractures with dorsal angulation but no comminution. On the other hand, Type 2-C fractures are extra-articular with dorsal angulation and displacement, yet no comminution. Lastly, Type 2-E fractures are intra-articular fractures with dorsal angulation and comminution. Figure 2.4 shows the schematic drawings for this classification method.

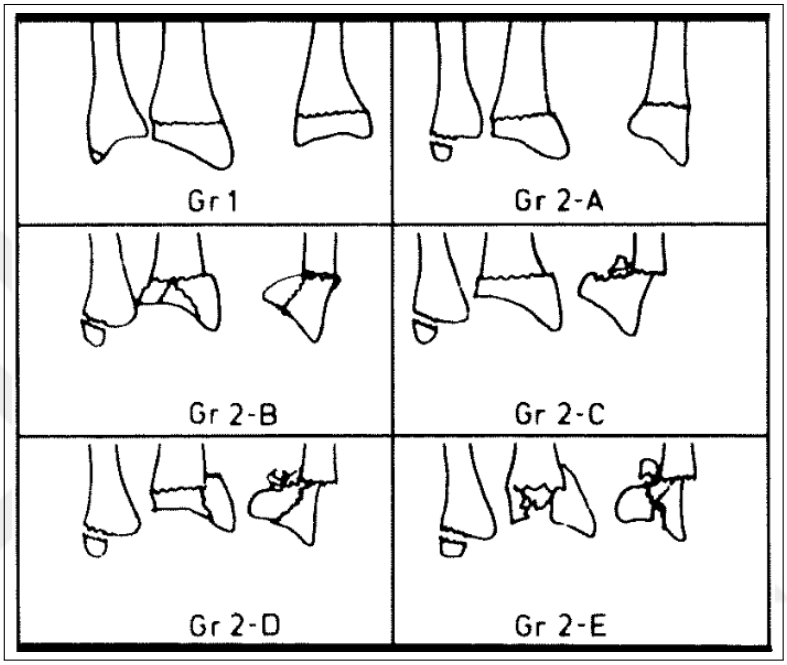


Figure 2.5. Lidström classification schematic drawing [20].

Older [23] classification system considers displacement and comminution aspects of the fractures to organize them into 4 groups. Type 1 category comprises fractures with dorsal angulation equal to or smaller than 5 degrees and radial length equal to or larger than 7 millimeters. Type 2 fractures have a dorsal angulation larger than 5 degrees and have a radial length between 1 and 7 millimeters. In contrast, Type 3 fractures involve severe comminution and have a radial length smaller than 4 millimeters with minimal displacement. Last but not least, Type 4 fractures have extensive comminution and a radial length of negative. Figure 2.6 depicts the schematic drawings for Older classification technique.

Last but not least, the Frykman [24] classification introduced in 1967 categorizes fractures into 8 types based on joint involvement. Type 1 fractures are extra-articular fractures which are fractures that are not related to joints. Type 2 fractures are also extra-articular with distal ulna fracture, which is the lower end of one of the two long bones in the forearm. Type 3

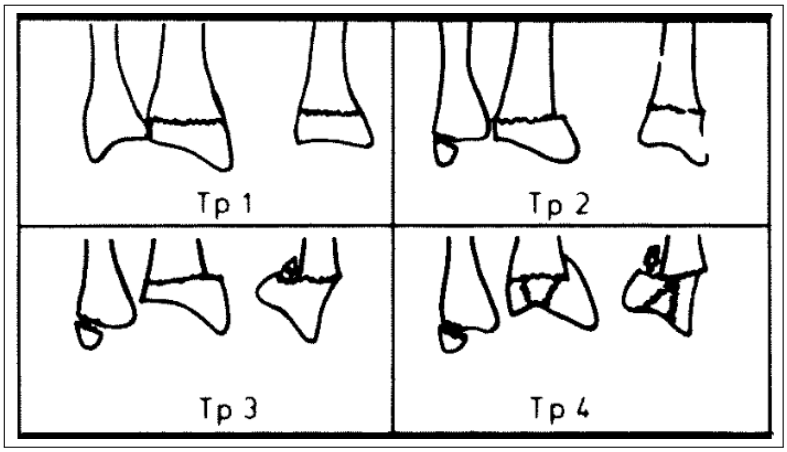


Figure 2.6. Older classification schematic drawing [20].

fractures only involve wrist joints while type 4 fractures also include distal ulna fractures as well. Type 5 fractures ensue a fracture in the distal radioulnar joint, which is a joint located in the wrist. Type 6 fractures have both distal radioulnar joint and distal ulna fracture. Type 7 fractures, on the other hand, have fractures in both radiocarpal and distal radioulnar joints. Lastly, type 8 fractures include radiocarpal, distal radioulnar, and distal ulna fractures. Figure 2.7 depicts the schematic drawings of type 1, type 3, type 5, and type 7 fractures of the Frykman classification system.

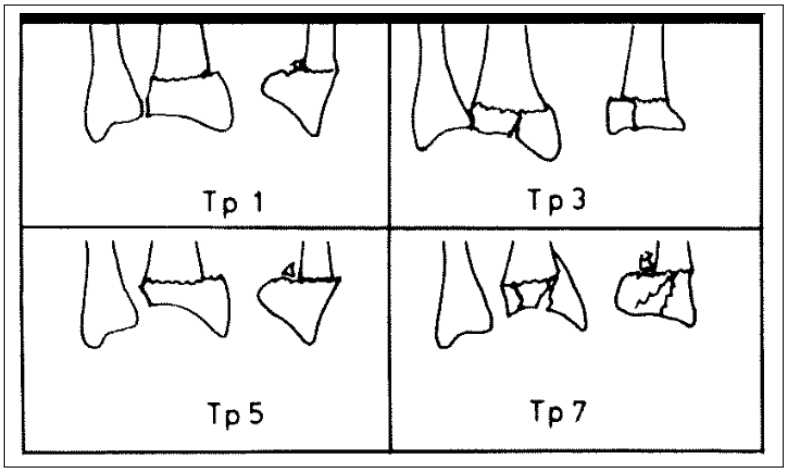


Figure 2.7. Frykman classification schematic drawing [20].

In a study, Solgaard [20] compares the previously mentioned five classification systems, namely Nissen-Lie, Gartland and Werley, Lidström, Older, and Frykman in terms of their prognostic value. It is claimed in the study that the classification system proposed by Older et al. [23] is the most effective for both predicting and achieving the likelihood of

anatomical reduction. Furthermore, this classification system's emphasis on comminution, which is the extent of fragmentation of the fractured bone, rather than focusing exclusively on the location of the fracture, makes it especially beneficial for clinical practice. In the study, it is explained that even though dorsal angulation is associated with the severity of the fracture, its effect on the outcomes of the DRF is greatly indirect. It is also stated that despite the fact that Nissen-Lie and Lidström's systems demonstrate practicability, they do not have a thorough consideration of the displacement and comminution present in the Older classification method. On the other hand, Frykman's system, in spite of its detailed focus on the joint involvement of a fracture reveals less predictive outcomes compared to Older's approach.

Plant et al. [13] evaluate the dependability and reliability of the AO classification system for DRFs. The study analyzes 456 radiographs of DRFs that are appropriate for operative fixation, which is a surgical intervention to treat fractures. Their key findings include that while the AO system is able to provide detailed information about fracture patterns, it lacks consistency when it comes to its group and subgroup classifications. Furthermore, it states that this classification system is ineffective at taking clinically relevant factors into account, such as the mechanism of injury or soft-tissue damage. Those points are highlighted to be vital for treatment planning.

2.4. TREATMENT OF DRFS

The treatment of DRFs is a complex process and the primary purpose is to avoid long-term complications while restoring wrist functionality. The treatment procedure is a spectrum of approaches that can range from more traditional methods, such as casting and splinting to surgical operations. The choice of treatment can depend on various factors; such as fracture severity, patient age, general health etcetera. Thus, this section explores various treatment techniques and the outcomes associated with different approaches.

Missakian et al. [25] investigate the effects of open reduction and internal fixation (ORIF) in treating 32 intra-articular DRFs. ORIF is a common surgical intervention to treat complicated bone fractures [26]. It is often performed on fractures that cannot be appropriately aligned

through more conservative approaches like casting. The process involves two primary steps: open reduction and internal fixation. For the open reduction stage, a surgeon makes an incision at the fracture zone to realign the bone fragments and restore the bone to its proper anatomical position. For the internal fixation phase, after the realignment of the bone, the surgeon secures the bone fragments in order to stabilize the fracture using a medical device, which can be metal plates, screws, or rods. During the recovery phase, these internal medical devices hold the bone in place to ensure proper realignment.

Missakian et al. review 32 patients where 17 of them are male and 15 are female. The patients' average age is documented to be 40 and the mechanism of most injuries is high-energy trauma, such as falls or vehicle accidents. In the study, the authors use the Frykman classification system discussed in the previous section. It is noted that most of the fractures entangled significant displacement and articular involvement.

Based on the type of fracture identified using the above method, an appropriate ORIF procedure is performed and internal devices are utilized for the fixation process, including wires and plates. Furthermore, in 50% of the cases, bone grafting is conducted to mitigate bone loss.

Approximately 54 months later a follow-up is conducted where it is seen that 88% of the patients achieved good to excellent outcomes. Furthermore, patients were able to retain about 80% of normal wrist motion and 73% of their grip strength. The patients are examined radiographically and the results demonstrated that fractures showed significant improvement in radial shortening and angulation.

In the study, ORIF is also compared with external fixation. The results reveal that ORIF shows similar and under certain circumstances better outcomes based on precise joint alignment and functional recovery.

Similarly, Abromo et al. [27] adopt and review the results of a treatment protocol on 581 patients. To evaluate the effectiveness of the protocol, the Disabilities of the Arm, Shoulder, and Hand (DASH) measurement is utilized.

DASH [28] is a evaluation of questionnaire developed to measure physical function as well as symptoms related to the upper limbs. Its essential purpose is to evaluate the impact of

upper extremity musculoskeletal conditions on daily life activities. Furthermore, it also aids in measuring the severity of the symptoms. Consequently, it helps medical practitioners to assess the outcomes of a treatment and monitor the recovery process. The overall score of a DASH assessment can range from 0 to 100. While higher scores indicate greater disability or worse outcomes, lower scores imply better results. Equation (2.1) shows the formulation to calculate the DASH score.

$$DASHScore = \frac{(SumofItemScores - 30)}{120} x100 \tag{2.1}$$

Abromo et al. categorize fractures based on their severity and reducibility. In the treatment protocol, undisplaced fractures are treated conventionally with casts. On the other hand, displaced or unstable fractures are treated with closed reduction, external fixation, or open reduction and internal fixation. The treatment protocol also considers the ages of the patients and their functional demands.

Their key findings of patient outcomes include that there is a significant improvement over time based on DASH scores and most patients are able to return to their near-baseline function within a year. Nonetheless, the median DASH scores decreased from 18 to 7.5 in nine months. Contrarily, operated fractures reveal worse initial outcomes, however, they are still comparable to the long-term results of non-operated fractures. In the study, it is also highlighted that older age and only reduction requiring non-surgery fractures are associated with lower DASH scores.

From a similar perspective, Trumble et al. [29] discuss the influence of fracture severity on functional recovery after surgical intervention. 43 patients with intra-articular DRFs are treated with ORIF and external fixation and the outcomes are reviewed.

The findings of the study included that the grip strength averaged 69% of the contralateral side, and the wrist range of motion was 75% of the uninjured side. Furthermore, pain relief scores demonstrate substantial improvement as most of the patients declared minimal or no pain during daily activities. The patients are examined radiographically and results revealed that there is a substantial improvement in fracture alignment. In conclusion, the study recommends aggressive surgical interventions for DRFs with articular incongruity surpassing

1 millimeter.

Ring et al. [30] evaluate the use of a different plate design for dorsal fixation of complicated DRFs. In the study, 22 patients are treated to evaluate the safety and effectiveness of the plate. It is highlighted that the plate is shaped like the Greek "π" and is specifically contoured to align with the anatomy of the distal radius. According to the AO classification, only Type 2 and Type 3 patients are involved in the study. The average patient age is 42 years, with injuries resulting mostly from high-energy trauma. The surgery process involves dorsal approaches with the preservation of tendons and in cases of severe comminution, external fixation is combined with the plate to sustain reduction.

On the 14-month follow-up, radiographic imaging is conducted on the patients and outcomes show that there is a high correlation of dorsal tilt and radial shortening. Functional results are rated as excellent or good in 13 of 22 patients using modified scoring systems. However, out of 22 patients, 5 patients are reported to develop tendon irritation. Consequently, out of these 5 cases, 4 of them required plate removal.

As a conclusion, Ring et al. reveal that while the plate is effective for stabilizing difficult DRFs and minimizing the necessity for external additions like casts, it comes with challenges related to tendon problems.

Cherubino et al. [31] provide an overview of DRF management, diagnostic criteria, treatment protocols, and their outcomes. This study has a particular emphasis on the requirement of individualized treatment based on various attributes of the DRF.

The fractures are evaluated in five parameters, namely (i) fracture type by using the AO classification system, (ii) the reducibility of the fracture, (iii) the reduction acceptability criteria, (iv) instability parameters, and lastly, (v) the involvement of the joint. The reducibility of the fracture criteria is decided by maintaining a radial height, inclination, and volar tilt. Instability factors involve dorsal angulation value to be higher than 20 degrees, there should be comminution present and the radial shortening must be larger than 4 millimeters.

Treatment alternatives are classified into four classes. The first one is closed reduction and cast immobilization, which is appropriate for stable extra-articular fractures. However, it is noted in the study this approach carries a high risk of a secondary displacement, especially

if the fracture is incorrectly classified in the previous phase. The second approach is closed reduction with percutaneous pinning, which can be used for extra-articular fractures with no comminution. This system is documented to achieve high stability and is minimally invasive. The third method is external fixation and it is suggested for fractures with metaphyseal comminution which is a radiographic criteria used for predicting DRFs instability [32]. This approach is linked with a moderate risk of complications. The fourth technique is the internal fixation of the fracture using volar plates which is recommended for more complex and unstable fractures. They can allow early mobilization and they show superior outcomes in terms of both anatomical and functional results. This technique is found to be especially effective for elderly patients as well as those who have osteoporosis.

In the study, different protocols are suggested for different fracture types, with more complex DRFs necessitating a combination of treatments. The functional outcomes of the fractures are measured using DASH and Gartland and Werley scoring systems. It is found that early mobilization and accurate reduction are correlated to better outcomes. However, some residual disabilities can still be observed in more severe cases.

From another perspective, Azad et al. [4] dissect the treatments and the complications associated with them across multiple age groups from 2005 to 2014. In the study, it is highlighted that DRFs follow a bimodal distribution, meaning that they have two peaks of occurrence in the population. As shown in Figure 2.8, one peak is seen among pediatric patients aged 10 to 14 years, while the second occurs among older adults aged 60 and above.

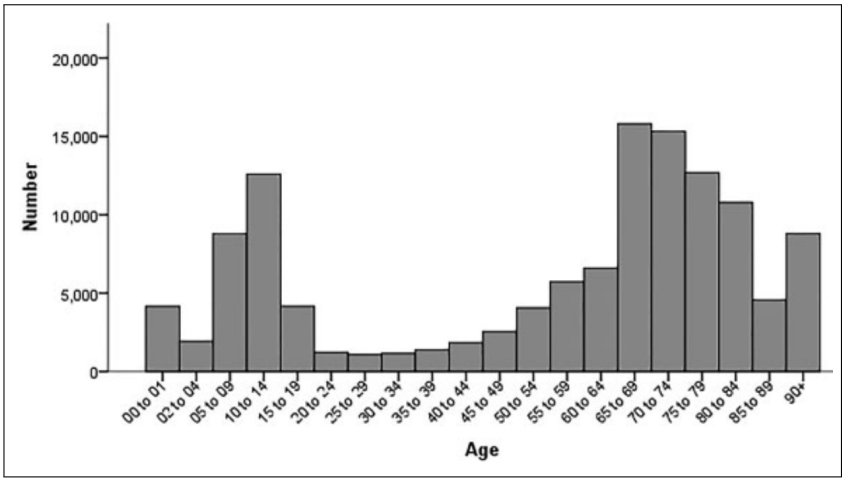


Figure 2.8. Incidence distribution of DRF across age [4].

The second peak is particularly pronounced in patients over 65 years, who exhibit a markedly higher incidence rate. Notably, while DRFs in children are more common in males, fractures in the elderly predominantly affect females, as depicted in Figure 2.9. This gender disparity is largely attributed to the higher prevalence of osteoporosis among women.

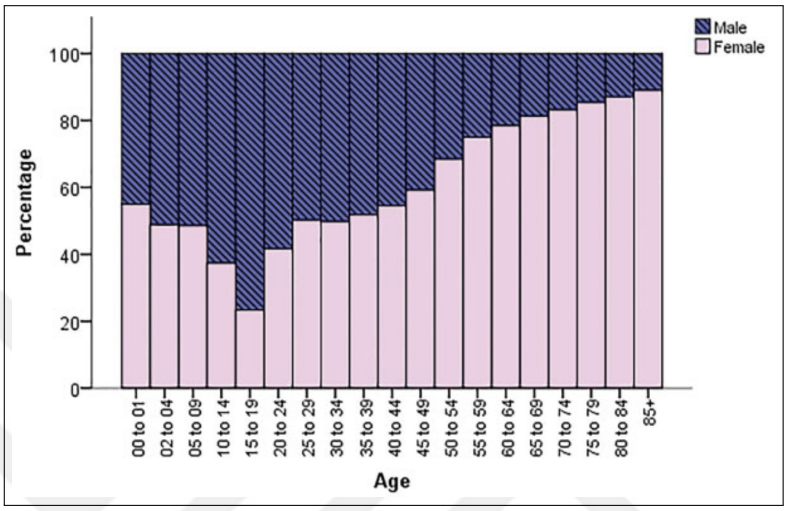


Figure 2.9. Incidence distribution of DRF across gender and age [4].

Figure 2.10 depicts the changing trends in the treatment of DRFs from 2005 to 2014. The most common approach across all age groups is non-operative treatment, which accounted for 86% of cases in 2005 however has declined to 77.5% by 2014. On the other hand, the internal fixation technique, especially with volar plates has increased significantly. Lastly, percutaneous fixation showed a decreasing trend across most age groups, except for pediatric patients.

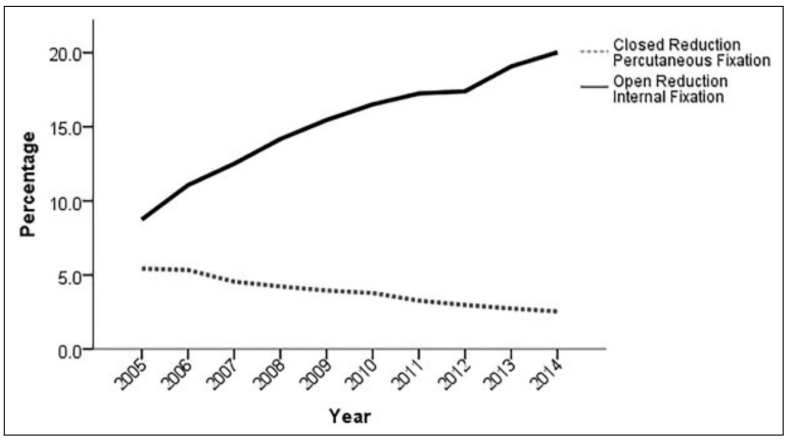


Figure 2.10. Yearly trends of treatment for DRFs [4].

For elderly patients older than 65 years, the shift toward internal fixation is apparent, which is attributed to the inherent instability of osteoporotic fractures. Likewise, internal fixation has also revealed an increase in the younger population, due to functional demands.

Azad et al. also investigates complication rates. According to their findings, the overall complication rate is documented to be 8.3% in 2005, with higher rates associated with surgical interventions. The most common complications are listed to be infections, tendon ruptures, and contractures. While internal fixation allows for early mobilization and independence, concerns related to healthcare costs are also addressed, particularly for elderly patients. They emphasize the requirement for a balanced approach to maximize functional recovery while keeping the financial burden to a minimum.

Last but not least, Perugia et al. [33] investigate whether it is necessary to restore anatomical parameters in DRFs that are treated with volar plate fixation. The objective of the study is to determine if restoring normal radiographic anatomical parameters is actually vital to achieving satisfactory functional outcomes. 51 patients treated with volar-locked plates are analyzed and their radiographic specifications are assessed for volar tilt, radial inclination, and ulnar variance. The measured functional outcomes include wrist range of motion, grip strength, and DASH.

Their findings show that volar tilt and ulnar variance are the most significant parameters to ensure full functional recovery. On the other hand, small variations of radial inclination do not critically impair the results. Lastly, stable internal fixation is found to contribute to improved recovery as it enables early rehabilitation. The majority of patients that participated in the study recovered full or near to full range of wrist motion with no substantial differences to the healthy hand and the average DASH score is documented to be 12.2 out of 100 which reflects minimal problems in conducting daily activities.

2.5. FRACTURE DETECTION WITHOUT ARTIFICIAL INTELLIGENCE

To accomplish effective diagnosis and treatment planning, accurate detection of DRFs is of utmost importance. Prior to the rapid developments in artificial intelligence domain and advances in computational tools, the detection and evaluation of DRFs relied primarily

on clinical examination and radiographic imaging. This section explores the approaches used in fracture detection without machine learning, deep learning, or artificial intelligence involvement.

Augat et al. [34] focus on investigating the radiological and biomechanical aspects of distal radius fractures. The study employs quantitive computed tomography (QCT) and dual X-ray absorptiometry (DXA) specifically for diagnosis. QCT is a special imaging tool that measures bone mineral density (BMD) in patients and it can provide precise information about the strength and geometry of the bone structure [35]. Similarly, DXA is also a non-invasive imaging technique for measuring BMD [36]. While DXA is considered to be the gold standard for diagnosing osteoporosis and assessing fracture risk, it provides a two-dimensional assessment of bone density and typically focuses on key fracture-prone areas such as the hip, spine, and forearm. On the other hand, QCT is able to deliver three-dimensional imaging of bone structure. While QTC's ability to offer detailed three-dimensional imaging makes it a valuable tool, it is less widely available, more expensive, and exposes patients to higher doses of radiation.

Agnus et al. also examine the relationship between BMD and the fracture strength of the distal radius using these two similar yet distinct imaging techniques. Their findings reveal that DXA is less accurate in predicting fracture loads compared to QCT. Furthermore, the study also highlights that the mechanism of injury also influences fracture patterns. Fractures produced with the wrist extension are found to be primarily caused by concentrated pressure on joints. However, neutral wrist positioning resulted in more evenly distributed pressures and varied fracture patterns. The clinical implications of this study highlight the importance of cortical bone parameters in assessing fracture risk. Measurement techniques such as QCT offer more detailed evaluations of bone strength compared to DXA.

With a similar perspective, Leerdam et al. [37] investigate the role of computed tomography (CT) imaging in detecting distal radioulnar joint (DRUJ) instability in 46 patients who have sustained a distal radius fracture and are being treated traditionally. DRUJ is the joint between the distal ends of the radius and ulna bones in the forearm. It plays a crucial role in wrist and forearm function, enabling rotational movements like pronation which refers to turning the palm down, and supination, which refers to turning the palm up.

DRUJ instability evaluation is performed using both CT and stress as well as a clunk test. The stress test's purpose is to assess static DRUJ instability, where unnatural movement occurs without forearm rotation. The patient's hand is held in both pronated and supinated positions. Thereafter, the examiner applies stress by manually manipulating the distal ulna in backward and volar forward directions. Finally, the movement of the wrist is compared to the movements of an uninjured wrist. On the other hand, the clunk test's objective is to evaluate dynamic DRUJ instability, which happens during rotational movements of the forearm. The examiner rotates the patient's forearm through pronation and supination and palpates for a "clunk" or abnormal movement during the rotational motion.

Furthermore, Leerdam et al. evaluates DRUJ instability by comparing clinical tests with CT findings. DRUJ instability is identified clinically in 35% of cases with the stress test and 15% with the clunk test. Furthermore, on CT, DRUJ instability is detected in 29 out of 46 patients, which equates to 63% of all cases. Thus, the correlation between clinical tests and CT findings is said to be moderate at best and often poor. The conclusions involve that the standardized criteria for diagnosing DRUJ instability on CT are lacking and observer-dependent variations in CT impact dependability. Thus, both clinical and radiological methods need standardization to achieve reproducibility to reliably diagnose instability.

In another study, Ganesh et al. [38] discuss the effectiveness of the dorsal tangential view (DTV) compared to CT for detecting prominent screws during volar locked plating of distal radius fractures. DTV is a specialized fluoroscopic imaging technique that is used during surgery to detect hardware-related complications [39]. The study especially focuses on complications that involve the penetration of the screw while fixing the DRF with a volar locking plate. DTV delivers a detailed view of the radius as it enables surgeons to identify screws that might poke beyond the bone and potentially damage extensor tendons.

The study involves 26 patients with 30 DRFs which are treated with ORIF using volar locking plates. Results reveal that while standard fluoroscopy detected prominent screws in 3 of 175 screws, DTV identified additional 10 prominent screws missed by the previous approach. Finally, postoperative CT detected 5 more screws that were not identified by either fluoroscopy or DTV. Thus, CT is discussed to be the most sensitive option for detecting screw penetration,

especially in critical compartments of the wrist. However, it is recommended that surgeons combine intraoperative techniques with CT imaging.

Ko et al. [40] investigate the accuracy of ultrasound compared to radiographs, for detecting DRFs in pediatric patients aged 2–15 years. 51 patients are involved in the study with a mean age of 9.9. The ultrasound procedure is conducted bedside by doctors and an uninjured patient is used as a control. Furthermore, X-ray images are taken in three views, namely anteroposterior, lateral, and oblique, and they are interpreted as the reference standard because X-ray is the gold standard for the detection of DRFs.

Of the two approaches evaluated, the ultrasound approach achieved a sensitivity of 97.1%. Only one discrepancy is noted between radiographic images and ultrasound results and upon follow-up, ultrasound diagnosis is validated. These findings emphasize that ultrasound can be referred to as a reliable diagnostic tool for DRFs in children, especially because of its advantages such as lack of radiation, quickness, cost-effectiveness, and patient comfort. However, limitations also include that there is a lack of standardized protocol for evaluating DRFs with ultrasound and this may limit reproducibility across different clinicians and settings.

Hacihaliloglu et al. [41] propose a new method for extracting bone surfaces and detecting fractures using 3D ultrasound imaging. The objective of the study is to address the challenges of ultrasound utilization for bone segmentation and DRF detection by leveraging three-dimensional local phase features while overcoming typical artifacts such as speckle noise or shadowing, which are commonly encountered problems that reduce the clarity of ultrasound images.

Using 3D Log-Gabor filters, the two-dimensional symmetry analysis method is converted to a three dimensional one to focus on the unique shapes and patterns of bones in the images. Their model is tested in three different settings, namely fake bones (phantom models), animal bones (ex vivo), and human tests (in vivo). For phantom models, plastic replicas are utilized to test how well their method detected fractures. For ex vivo testing, porcine specimen specimens are employed to test the model on real bones covered with muscles. Lastly, for in vivo testing, human wrists are scanned to test if the approach functions in real-life scenarios. Consequently, it is found that the model successfully identifies the bone surfaces

with an error rate of less than 0.62 millimeters. Furthermore, their approach can find small gaps between broken pieces with an error degree of less than 0.65 millimeters which means that the three-dimensional method creates clearer and smoother images compared to earlier two-dimensional approaches.

In another study, Hacihaliloglu et al. [42] build upon their preious study [41] by refining and expanding their methods for using ultrasound to detect bones and fractures. With a similar goal in mind, the objective is to improve ultrasound imaging such that it reliably detects bones and fractures in 3D since traditional methods like CT scans involve radiation and may not always be available during surgeries.

Once more, a more advanced and more precise method to process ultrasound images using 3D Log-Gaber filters is developed. The model's testing is conducted on three different settings, identical to the previous study, phantom models, ex vivo, and in vivo. In this study, both the bone surfaces and gaps between bone fragments were identified with an error rate of less than 0.6 millimeters. Compared with the previous study, the processed ultrasound images are noted to be much clearer and smoother. Furthermore, the proposed method performed well even when the bone is covered by soft tissue or the ultrasound beam is not perfectly aligned, which makes ultrasound a strong alternative to CT scans and fluoroscopy.

2.6. AI-ASSISTED DISTAL RADIUS FRACTURE STUDIES

Recently, artificial intelligence (AI) has undergone significant advancements, establishing itself as a powerful tool across a wide range of fields. Medicine is one of the key domains where AI has demonstrated substantial contributions, particularly through automated, accurate, and rapid fracture detection using advanced computer vision algorithms and deep learning models to analyze medical images. AI has transformed various aspects of healthcare, from diagnostic imaging to predictive analytics and personalized treatment. In this context, this section focuses on key advancements in AI-assisted distal radius fracture detection.

Shen et al. [43] conduct a review of studies to examine the transformative impact of deep learning on medical imaging, with a particular focus on its role when it comes to automating feature extraction and diagnostic accuracy improvement. It is highlighted

that deep learning approaches enable hierarchical feature representations directly from raw data, unlike traditional methods that depend on handcrafted features. Consequently, it reduces the dependency on domain-specific expertise. However, key challenges remain, including the limited availability of labeled datasets, lack of interpretability, and difficulties with domain adaptability. The study concludes with several recommendations for future research: integrating domain knowledge, improving model generalization, and addressing the 'black-box' nature of these models.

From a similar perspective, Lindsey et al. [44] evaluate the possibility of deep neural network (DNN) enhancing the ability of clinicians to detect fractures from X-ray images. A deep convolutional neural network (DCNN) is trained on a dataset that consists of 135,345 labeled X-ray images of different body parts and the model is optimized for image interpretation tasks common in radiology. The trained network is then tested on separate, never-before-seen X-ray images. Then, the same X-ray images are interpreted by radiologists and orthopedic surgeons interpreted the same radiographs with and without AI assistance. The results are measured using metrics such as sensitivity, specificity, and area under curve (AUC). The findings reveal that the DCNN model established a high standalone accuracy as AUC exceeds 0.9 in detecting fractures. Moreover, sensitivity and specificity metrics indicated robust performance, especially for complex or subtle fractures. In conclusion, AI assistance did improve the accuracy of clinicians as the DCNN is effective in reducing false negatives and provides quick, interpretable results that could be incorporated into clinical workflows. Figure 2.11 shows a graph of clinician assessment accuracy across reading time with and without the help of AI. Each point in this figure represents a cluster of radiographs containing one-tenth of the testing X-ray images. The AI-assisted system has provided better accuracy in readings along with speed, especially in more complex fractures.

In line with the previous study, Blüthgen et al. [45] compare the detection and localization of DRFs conducted by radiologists versus a deep learning system. The system is trained with 524 wrist X-rays, internally tested with 100 images, and externally validated with 200 images from a different dataset. Similar to the previous study, radiologists interpreted the test radiographs with and without the contribution of AI. Afterward, AUC, sensitivity, and specificity metrics are calculated for both the deep learning system and the radiologists. Furthermore, heatmaps are produced to depict the radiologists' diagnosis and the deep

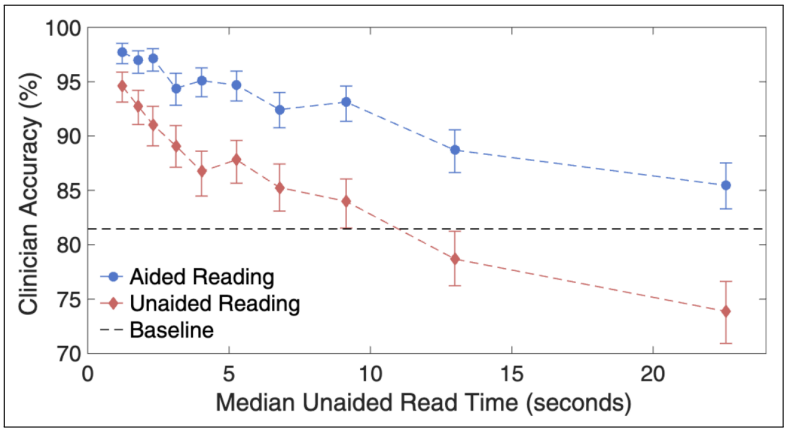


Figure 2.11. Average time and accuracy of reading a radiograph [44].

learning system's findings. Figure 2.12 shows a sample heatmap structure where "a" indicates the original X-ray image, "b" reveals the heatmap produced by the deep learning system, "c" exhibits contours describing the 2 radiologists' region of interest where the orange annotated area corresponds to their mutual region of interest, and "d" displays the correctly localized fracture marked with yellow, along with the radiologists' contour lines. Thus, the deep learning system's ability to generate heatmaps that can correctly highlight the location of fractures is verified.

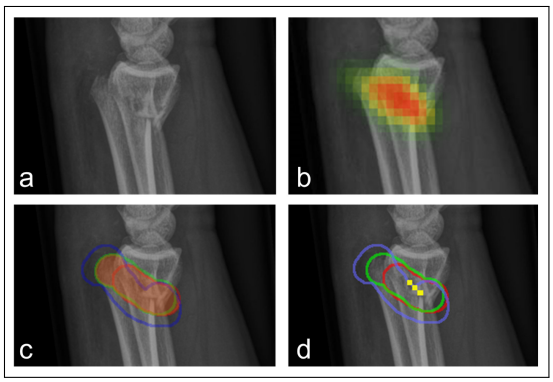


Figure 2.12. Heatmaps generated by the system versus radiologists [45].

On the internal test data, the deep learning system reached an AUC of approximately 0.945. However, on the external test data, the AUC metric is lower: at around 0.895, which indicates reduced generalizability. While radiologists performed comparably to the deep learning system on the internal test set, they outperformed it on the external dataset. Nonetheless, the deep learning system is noted to be particularly effective in aligning its heatmap predictions with radiologists' fracture annotations, achieving over 90% agreement. Limitations of the

model include difficulties in detecting highly displaced fractures and rare fracture patterns. Its performance on external images further emphasizes the need for representative and diverse training data.

A study by Zhang et al. [46] aims to develop and evaluate a deep learning ensemble model for detecting DRFs from X-ray images and compare its diagnostic accuracy, sensitivity, and specificity with that of orthopedic surgeons and radiologists. The researchers collect a dataset that consists of radiographs from 3,240 patients where 1,620 include fractures and 1,620 of them does not. The images are split into two classes based on their views, namely anteroposterior and lateral. A total of 6,536 images are then divided into three sets, training, validation, and testing, 70%, 15%, and 15% respectively. An ensemble deep learning model is trained that combines three state-of-the-art backbone architectures, which are RetinaNet, Faster R-CNN, and Cascade R-CNN. While RetinaNet is a one-stage model, providing fast and efficient detection for objects of different scales, Faster R-CNN is a two-stage model that generates region proposals for object detection and classification. Last but not least, Cascade R-CNN is a multi-stage model and aids in improving the accuracy of the model by conducting iterative refinement of predictions. Figure 2.13 shows the comprehensive diagnostic process of the ensemble model for DRFs using both anteroposterior and lateral X-ray images of the same wrist.

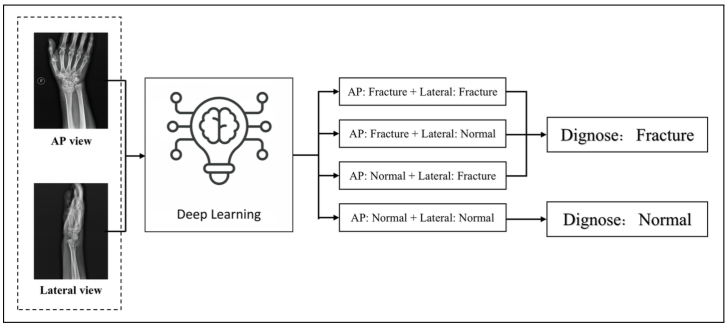


Figure 2.13. Diagnostic process of ensemble model [46].

The results reveal that the ensemble model achieved 97.03% accuracy, 95.7% sensitivity, and 98.37% specificity. Furthermore, orthopedic surgeons with at least 3 years of experience achieved an average accuracy of 93.69%, sensitivity of 91.94%, and specificity of 95.44%. Lastly, radiologists achieved an average accuracy of 92.53%, sensitivity of 90.44%, and specificity of 94.62%. In conclusion, the ensemble model outperformed both groups across

all metrics. Furthermore, the model provides red-box heatmaps on radiographs to enhance interpretability and diagnostic efficiency.

Adopting a similar view, Hardalaç et al. [47] aim to develop and assess the accuracy, recall, and precision metrics of deep learning models using advanced single-stage, two-stage, and ensemble models. The creation of the dataset involved the gathering of X-rays from 542 patients with DRFs. The data is then split into 80% training, 10% validation, and 10% testing sets. On the mentioned dataset, a total of 10 object detection models are trained. Single-stage models included RetinaNet, Feature Selective Ancher-Free (FSAF), and Probabilistic Anchor Assignment (PAA). On the other hand, two-stage models are listed to be Faster R-CNN, Dynamic R-CNN, and Deformable Convolutional Networks (DCN). Last but not least, 5 ensemble models are assembled by combining results from the previously mentioned one-stage and two-stage models using Weighted Boxes Fusion (WBF). An additional ensemble model is also trained using 5 other ensemble models, named Wrist Fracture Detection - Combo. Among the individual models, the PAA model reached the highest precision score of 0.754 and recall of 0.496 on the test set. Out of all ensemble models, the Wrist Fracture Detection - Combo model achieved the best metrics with an average precision of 0.86, irevealing a 10% improvement over the PAA model.

From a comparable viewpoint, Thian et al. [48] also train a CNN-based object detection model to automatically detect and locate fractures from X-ray images. A total of 7,536 wrist radiographs with anteroposterior and lateral views are collected from the emergency department for two years and the images are split into training and validation sets with a 9 to 1 ratio. The test set consists of external 524 radiographs. The fractures are annotated by radiologists using bounding boxes. The trained model employs the Faster R-CNN architecture with Inception-ResNet as the base CNN and the model is trained to identify fractures, provide classification as ”fracture” or ”no fracture”, and localization with bounding boxes. Sensitivity, specificity, and AUC metrics are calculated per radiograph. Furthermore, subgroup analysis is conducted to evaluate the performance of the model across pediatric and adult patients, images with casts, and maximally displaced and minimally displaced fractures. For frontal and lateral views combined, the model reached a sensitivity of 96.2%, a specificity of 84.45%, and an AUC of 0.92. The subgroup analysis revealed that no significant differences in model performance were noticed. However, due to artifacts caused by casts, a

lower specificity for radiographs involving casts is noted. Furthermore, higher sensitivity is acquired by the model for highly displaced fractures compared to minimally displaced ones.

From a different perspective, Yahalomi et al. [49] evaluate the feasibility of training a Faster R-CNN neural network to detect and locate DRFs utilizing a small dataset of radiographs to attain high diagnostic accuracy. The initial dataset included 38 X-ray images of DRFs and 40 X-ray images without fractures. After various augmentation steps are applied to the dataset, such as mirroring, contrast, and brightness, which expanded the training and testing set to 4,476 images. The Faster R-CNN framework with a Visual Geometry Group - 16 (VGG-16) backbone is employed for the detection of fractures. Transfer learning is performed by utilizing the pre-trained weights from the ImageNet dataset. For the image preprocessing step, all radiographs are resized to a higher resolution to detect smaller fractures productively.

The model is trained with a batch size of two and a learning rate of 0.0003. Moreover, various hyperparameters are fine-tuned to improve precision. The study used mean Average Precision (mAP), which is a key metric in object detection, and overall classification accuracy for evaluation. While the model acquires a classification accuracy of 96% for fracture detection, the mAP score is documented to be 0.86 despite being trained on a small dataset. The findings show that the model can demonstrate robust performance, underlining the potential of Faster R-CNN for limited data. However, the study's heavy reliance on image augmentation to compensate for the small size of the dataset can be seen as disadvantage real-world scenarios.

In line with the previous study, Tobler et al. [50] assess the effect of training set size by training a DCNN model for detecting DRFs, metal, and casts from X-ray images. A total of 15,775 radiographs are collected and classified into three classes, namely fracture detection, fracture classification, and metal or cast detection. While fracture detection identifies the absence or presence of fractures in a given radiograph, fracture classification categorizes fractures based on characteristics such as displacement, joint involvement, and bone fragmentation. Lastly, in this study, metal and cast detection step recognizes the presence of surgical hardware or casts. The data is split into training, validation, and testing sets, with subsets ranging from 500 to 9,000 images to evaluate the effect of training set size. The Residual Network 18 (ResNet-18) architecture is employed for training. Furthermore, stochastic gradient descent (SGD) is utilized for training with image augmentations.

Multiple DCNN models with different architectures are trained on various training set sizes. Larger datasets consistently improved performance for fracture detection and most classification tasks, except for joint involvement, where accuracy remained low regardless of set size. Among these, the ResNet-18 model trained on the full dataset achieved the best evaluation metrics, reaching an AUC of 0.98 with an accuracy of 93.8%. For comparison, radiology residents diagnosed the same test set and achieved an AUC of 0.98, indicating that ResNet-18's performance is comparable to human-level expertise. The model also showed excellent results in detecting metallic implants and casts, achieving AUCs between 0.99 and 1.00.

However, in more challenging fracture classification tasks, such as identifying multiple fragments, fragment displacement, and joint involvement, the model's evaluation metrics are notably lower. Specifically, for multiple fragment detection, accuracy was 78.2% with an AUC of 0.85. For fragment displacement, the AUC dropped to 0.74, which is significantly lower than the residents' AUC of 0.92. The joint involvement task yields the weakest results, with an AUC of 0.65 compared to 0.9 by the residents.

Suzuki et al. [51] aim to evaluate the diagnostic accuracy of DCNNs for DRF detection by comparing the model's performance to orthopedic hand surgeons. The authors collect a dataset comprising 503 radiographs with fractures and 289 without from four different hospitals. Consequently, the final dataset consisted of 905 anteroposterior X-ray images and 728 lateral radiographs. With this dataset, Efficient Network (EfficientNet) B2, B3, B4, and B5 frameworks are trained using transfer learning. Additionally, separate models are trained for anteroposterior and lateral views. Thereafter, the best-performing models are combined into an ensemble for multiview analysis and this process is noted to be advantageous for improving diagnostic performance as it enables the model to detect subtle fractures.

The ensemble model achieved an accuracy of 99.3%, a sensitivity of 98.7%, and a specificity of 100%. When the ensemble model's diagnostic performance is compared to that of 3 hand surgeons, the performance turned out to be equal or better. Not only that, but the surgeons had a slightly lower accuracy ranging between 94.7 and 97.3% with a lower sensitivity of 96% compared to the trained model.

Taking a different approach, Oka et al. [52] use biplane X-rays along with an AI system trained

on a relatively small dataset to diagnose DRFs. Their dataset consists of 369 anteroposterior and 360 lateral fracture radiographs as well as 129 anteroposterior and 125 lateral radiographs without fractures. Through data augmentation, the dataset is expanded to 3,245 images for training and validation. A CNN model with a backbone of VGG16 architecture is trained for fracture detection. While the CNN's dual-layer output system identifies the presence or absence of fractures, the diagnostic process follows a structured sequence: anteroposterior images aew analyzed first, and if no diagnosis could be established, the corresponding lateral image is subsequently reviewed. This stepwise diagnostic procedure is illustrated in Figure 2.14.

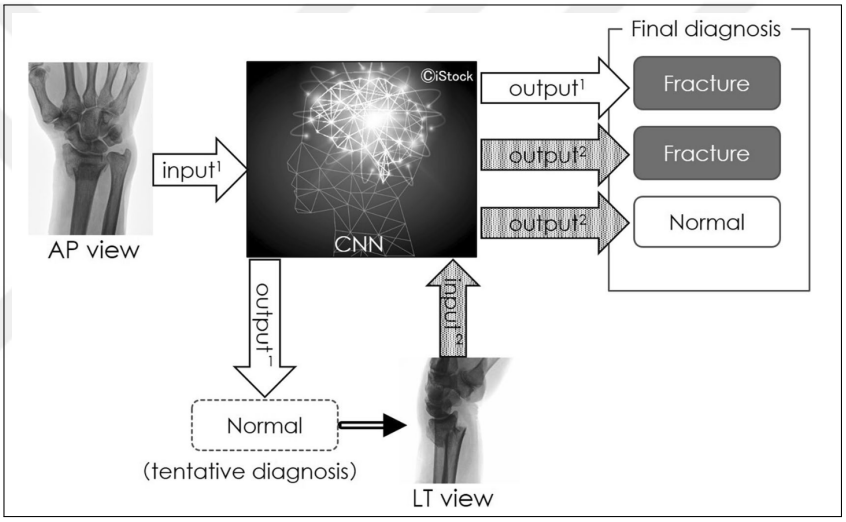


Figure 2.14. Diagnostic process of the CNN model [52].

As a result, their proposed model achieved an accuracy of 95.7%, sensitivity of 95%, specificity of 97.2%, and AUC of 0.99 for lateral images. Making use of anteroposterior and the lateral radiographs simultaneously, the model achieves better results: an accuracy of 98%, a sensitivity of 98.6%, a specificity of 96.7%, and an AUC of 0.991. The main contribution of the study is the use of biplane X-rays to improve the diagnostic reliability with high accuracy and sensitivity using a small dataset. However, dataset size can also be deemed as a limitation, as the study's dependence on a small dataset raises concerns about its generalizability. Additionally, the CNN is not able to localize the fractures, reducing the approach's interpretability.

Kim et al. [53] evaluate the diagnostic performance of Dense Network 161 (DenseNet) and ResNet-152 architectures for detecting distal radio-ulnar fractures in plain radiographs

from emergency room (ER) patients. While DenseNet-161 uses densely connected layers to propagate features across the network, ResNet-152 employs skip connections to learn residual features. 9,984 radiographs are collected from 2,609 ER patients with wrist trauma from 2018 to 2020. In total, the dataset included 4,851 images with fractures and 5,133 without fractures. The dataset is then split into training and testing sets with a 9 to 1 ratio. Contrast-limited adaptive histogram equalization (CLACHE) is utilized to enhance the images' qualities in the dataset. Additionally, images are resized to 550×660 pixels to reduce computational requirements. Both of the CNN architectures are pre-trained on ImageNet and are fine-tuned to conduct fracture detection. Gradient-weighted Class Activation Mapping (Grad-CAM) is utilized to highlight image regions contributing to CNN's decision in order to improve interpretability, as depicted in Figure 2.15.



Figure 2.15. Grad-CAM mapping on a sample radiograph [53].

DenseNet-161 surpassed ResNet-152 architecture across all evaluation metrics. While DenseNet acquired an overall accuracy of 90.3%, ResNet reached an accuracy of 88.5%. Most of the errors in predictions stem from artifacts related to imaging or old fractures.

From another interesting viewpoint, Anttila et al. [54] aim to detect DRFs with a deep learning model using a dataset with precise and pixel-level annotations. A total of 3,785 emergency wrist radiograph examinations are collected from six different hospitals, which included both posteroanterior and lateral views. Then, the radiographs are annotated pixel-level by physicians with an annotation tool the authors developed. A modified U-Net

architecture consisting of 25 layers with skip connections and upsampling is utilized for training. The model predicts fracture confidence values for each pixel in the X-ray images. For all radiographs, the model acquired an AUC of 95% and an accuracy of 91% for X-ray images without casts with a sensitivity of 94% and a specificity of 87%. The comparison of posteroanterior versus lateral view radiographs reveals that the model's fracture detection is higher in posteroanterior radiographs with an AUC of 0.97; whereas lateral views' AUC is 0.94. While false negatives are extensively observed in subtle fractures or due to imaging artifacts, agreement between posteroanterior and lateral views is documented to be 87%, with a low error rate for misclassification across both views.

Min et al. [55] introduce a two-stage ensemble deep learning framework using YOLOv5 and EfficientNet-B3 for automated classification of intra and extra-articular DRFs. A total of 400 radiographs are collected from 340 patients where 160 of the instances are intra-articular, 89 of them are extra-articular, and 51 are normal instances. Radiographs in the dataset only consist of posteroanterior view and they are annotated by orthopedic surgeons, split into training and testing sets. The first step of the two-stage framework involved region of interest (ROI) detection where an ensemble of the You Only Look Once version 5 (YOLOv5) model is employed to localize the region. Stage two involved the classification process where the EfficientNet-B3 model is ensembled into the YOLOv5 model to categorize fractures as intra or extra-articular. Lastly, the ensemble predictions are averaged to achieve the final classification detection. Figure 2.16 shows the process of detection for the ensemble model.

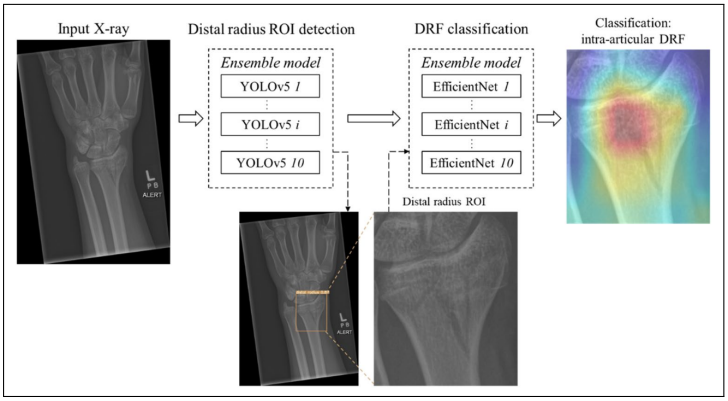


Figure 2.16. EfficientNet and YOLOv5 ensemble model process [55].

Evaluation of the ensemble model show that YOLOv5 achieved a true positive rate (TPR) of 100% for the ROI detection and an Intersection over Union (IoU) value of 0.81, which

indicates a high localization accuracy. The classification performance of the ensemble model reached an AUC of 0.82, an accuracy of 81%, a sensitivity of 83%, a precision of 90%, and an F1-Score of 0.86.

From a different perspective, Raisuddin et al. [56] assess the efficiency of a deep learning pipeline for detecting DRFs with a particular emphasis on problematic cases requiring CT verification. The dataset is divided into 3 distinct sets; namely training set, test set number 1, and test set number 2. The training set consists of 1,946 images where 953 of them involve fractures and 993 of them are healthy wrists. Test set number 1 involves 207 more general and simple cases. On the other hand, test set number 2 included 105 challenging cases that require CT confirmation due to diagnostic uncertainty. The deep learning pipeline, named DeepWrist in the study, consisted of 2 blocks. The first block -the ROI localization block- localizes key anatomical landmarks such as distal radius or ulna by utilizing an hourglass network where the ROIs are cropped for fracture detection. The second block -the fracture detection block- employs a CNN network that is fine-tuned to detect fractures from the cropped ROIs. On top of that, a multi-view ensembling is implemented to combine predictions from posteroanterior and lateral views to improve accuracy. Last but not least, Grad-CAM visualization is implemented to generate heatmaps to enhance the model's interpretability.

For simpler cases that do not require CT verification, the model acquired an AUC of 0.98 for the posteroanterior view and 0.98 for the lateral view. The classification performance of the model is evaluated using the area under precision-recall curve (AUPRC) which reached 0.99, indicating near-perfect classification performance. However, when compared with real-life physicians, radiologists slightly outperformed the DeepWrist pipeline in terms of sensitivity and specificity.

For more challenging cases requiring CT verification, the model achieved significantly lower metrics. While the AUC is 0.83 for lateral view radiographs, it drops to 0.81 for posteroanterior view images. AUPRC drops to 0.64 which underlines the need for additional refinement in detecting subtle fractures or atypical patterns. One of the most crucial findings of this study is that it highlights issues related to hidden stratification, where the model trained on general cases performs poorly on out-of-distribution challenging cases.

Kim and MacKinnon [57] focus on the implementation of transfer learning with pre-trained

CNNs to conduct automatic detection of DRFs from lateral X-ray images. The dataset is composed of 1,389 lateral radiographs where 695 of them are images with fractures. Through the use of data augmentation techniques, the dataset's final size is raised to 11,112 images. Moreover, an additional 100 X-ray images, half of which contain fractures, are used for final testing. To achieve transfer learning, the pre-trained Inception version 3 CNN architecture is employed. The top layer of the network was retrained to classify X-rays as "fracture" or "no fracture". During the training, the model converged with minimal overfitting as the augmentation step improved generalization by accounting for wrist size variation and positioning.

The newly trained CNN model achieved an AUC of 0.95 which indicated outstanding diagnostic performance, sensitivity of 90%, and specificity of 88% which demonstrates that transfer learning and augmentation could reach high accuracy levels despite the use of a small-sized dataset.

Daluiski et al. [58] developed a CNN-based model that both detects and localizes DRFs and compared the results to that of inexperienced clinicians, with an objective to assess whether the CNN could provide superior performance generating interpretable visual outputs to support clinical decision-making. In this study, a total of 36,408 wrist radiographs are utilized where 31% contain fractures. Initially, the dataset is divided into training, validation, and testing sets. Furthermore, an extra 2,215 X-ray images are utilized for testing purposes. The CNN is structured so that it performs semantic segmentation and treats the detection task as a pixel-based classification problem. In order to evaluate the comparison between the CNN architecture and clinicians, two radiology trainees, two physician extenders, and two practicing urgent care physicians participated. Each clinician reviewed 250 X-rays which are randomly sampled from the extra testing set.

On the test set derived from the initial dataset, the model acquired an AUC of 0.97. On the other hand, predictions on the external images revealed an AUC of 0.98 and a sensitivity of 95% with a specificity of 93%. Compared to the CNN model, inexperienced clinicians demonstrated significantly lower sensitivity and specificity. Moreover, the CNN is documented to identify the fractures more reliably, as well as localize them more precisely. The study highlights that the system can be useful in residency training, as it can assist trainees

in learning fracture detection. Furthermore, the system can be used to support urgent care and radiology settings with high diagnostic workloads.

Similarly, Knight et al. [59] aim to evaluate the feasibility, accuracy, and reliability of 2D and 3D ultrasound in diagnosing pediatric patients with DRFs. Moreover, the study evaluates the diagnostic performance of 7 clinicians, categorized by the authors as novice, intermediate, and expert based on their prior medical and ultrasound experience, by comparing their findings with the AI-integrated system. The novice readers had no medical education or ultrasound background, intermediate readers include one family/emergency medicine physician with limited POCUS experience and one undergraduate student, while expert readers consisted of a pediatric radiology fellow and a dual-fellowship trained pediatric musculoskeletal radiologist with 15 years of experience.

Accordingly, a total of 127 pediatric cases with wrist trauma are utilized in the study. Initially, 2D and 3D ultrasound scans of the wrists are acquired from various viewpoints. Then, ResNet34 and DenseNet 121 models are trained on the labeled ultrasound images. 7 human clinicians interpreted and diagnosed randomized, anonymized 2D and 3D ultrasound images which are then compared to ground truth.

Evaluation results show that the trained ResNet34 achieved an AUC of 0.93 for 2D ultrasounds as well as 3D ultrasounds. However, the model revealed a lower sensitivity of 82% for 2D ultrasounds but 100% for 3D ultrasounds. On the other hand, the DenseNet model's evaluation metrics surpassed those of ResNet, with an AUC of 0.94 with a sensitivity of 91% for 2D ultrasounds and 0.91 with 100% sensitivity for 3D ultrasounds. This indicates that both 2D and 3D ultrasounds are reliable for diagnosing DRFs.

The clinicians categorized as novice in the study achieved a sensitivity of 62% and specificity of 82% in detecting DRFs using 2D ultrasound images. On the other hand, when 3D ultrasound images are utilized, they reached a sensitivity of 63% with a specificity of 91%. The intermediate physicians achieved a sensitivity of 88% and specificity of 94% on 2D ultrasound images. On 3D ultrasound images, they attained a sensitivity of 88% with a specificity of 95%. Lastly, clinicians categorized as experts in the study acquired a sensitivity of 97% and specificity of 90% on 2D ultrasound images. On 3D ultrasound images, they reached a sensitivity of 98% with a specificity of 93%. Consequently, it is reported that the

AI performs comparably to experts, particularly with 3D ultrasound images, and succeeded in surpassing novice and intermediate clinicians across all ultrasound versions and metrics.

3. METHODOLOGY

This section provides a thorough overview of the methods and approaches employed in the development and assessment of AI-based systems for detecting DRFs. The datasets utilized throughout the literature are described initially, including preprocessing techniques and the annotation process. Afterward, the deep learning architectures and procedures are explained, including model details, hyperparameter tuning, and evaluation metrics necessary to evaluate the models. Finally, the steps for validating the model's performance and ensuring its clinical applicability are outlined.

3.1. IMAGING MODALITIES AND DATASETS

The first and most crucial requirement to develop a robust AI model is the availability of high-quality and diverse datasets. In the context of DRF detection, this may include radiographs, CT scans, medical reports, or ultrasound images. This subsection outlines the key elements of dataset preparation commonly utilized in this domain.

Datasets for distal radius fracture detection generally come from radiology departments, emergency rooms, or publicly available repositories. These datasets include various imaging modalities such as X-rays, ultrasound images, and in some cases, CT scans. Datasets used in fracture detection studies often consist of thousands of images and, when limited in size, are typically expanded using techniques such as image augmentation.

X-rays utilize electromagnetic radiation, which is the energy discharge consisting of electromagnetic waves to construct shots of the interior of the body [60]. X-ray photons are generated once high-energy electrons are collided with a metal target inside an X-ray tube which transforms electrical input power into X-rays. Thereafter, an X-ray beam passes through the body such that different tissues of the body absorb X-rays differently, which allows the variety between dense tissues and soft tissues to be observable. While dense tissues such as bones absorb more X-rays and appear white on the film, soft tissues such as muscles or organs absorb fewer X-rays and appear darker. Lastly, air-filled spaces, such as lungs absorb the least amount of beams, appearing the darkest. Once some of the X-rays pass through

the body, the remaining ones strike a digital sensor or film, consequently creating an image, revealing details about bones or organs to aid diagnosis. Figure 3.1 shows an example X-ray image where bones appear lighter and soft tissue appears darker.

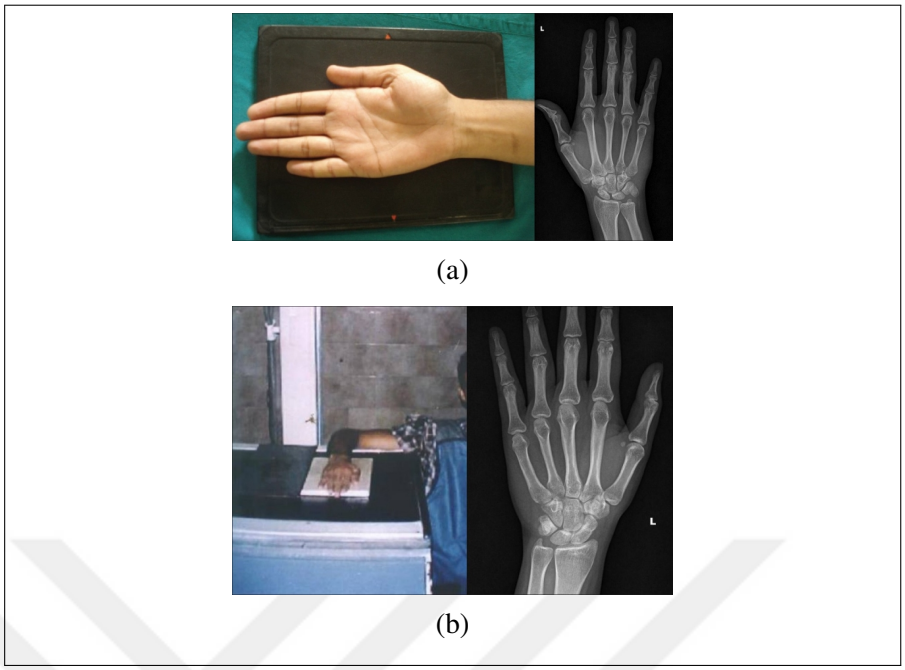


Figure 3.1. Sample X-ray image.

Depending on the part of the body that is being X-rayed, the patient either sits or stands. Then, the body part of interest is placed between the X-ray source and the detector. There are various angles in radiography for visualizing the complete anatomy of bones and two of the most typical imaging angles are posteroanterior and lateral views.

As shown in Figure 3.2a, in an anteroposterior view, the patient places their hand palm up onto a flat surface with their fingers extended and the X-ray beam passes through the hand from front to back [61][62][63], also known as anterior to posterior. The view is said to be posteroanterior if the patient's palm is facing downwards as illustrated in Figure Figure 3.2b. This view shows a frontal view of the anatomy, displaying the distal radius, ulna, carpal bones, and joint spaces in their natural position. It is beneficial for evaluating radial height and inclination, as well as the fracture alignment in the coronal plane.

In the lateral view, the X-ray beam passes through the body from side to side, which makes



(a)

(b)

Figure 3.2. Anteroposterior and posteroanterior views. (a) Anteroposterior angle, (b) Posteroanterior angle [61].

the position of the wrist perpendicular to the previous view. The detector is positioned to the side of the wrist, and the X-ray source is opposite it. The patient's elbow should be bent at a 90-degree angle such that a somewhat straight line can be drawn through the relative anatomy of the wrist. The hand should be placed with the thumb pointing upward, as presented in Figure 3.3, providing a side view of the wrist anatomy. In this view, wrist structures such as the radius, ulna, and carpal bones can be inspected, essential to asses the volar or dorsal displacement of the fractures, angulation and/or alignment of the bone fragments, and lastly, joint alignment in the sagittal plane.



Figure 3.3. Lateral view X-ray imaging [61].

Other imaging angles include oblique, scaphoid (ulnar deviation), carpal tunnel, and radial deviation views. While anteroposterior and lateral views deliver a lot of information,

additional views may help in characterizing fracture patterns better or may yield a comprehensive 3D understanding of the fracture. In an oblique view, the wrist should be rotated approximately 45 degrees relative to the X-ray beam, as depicted in Figure 3.4. On top of providing an intermediate angle of the distal radius and ulna, it also highlights different bones in the area more clearly. It is particularly useful for visualizing the alignment of fractures extending obliquely across the radius.

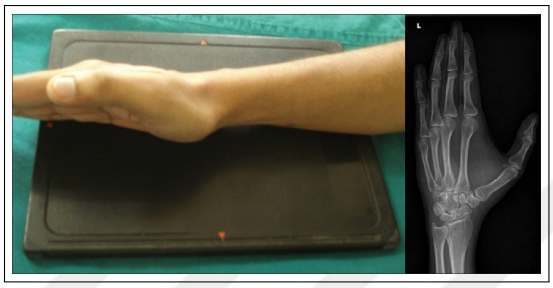
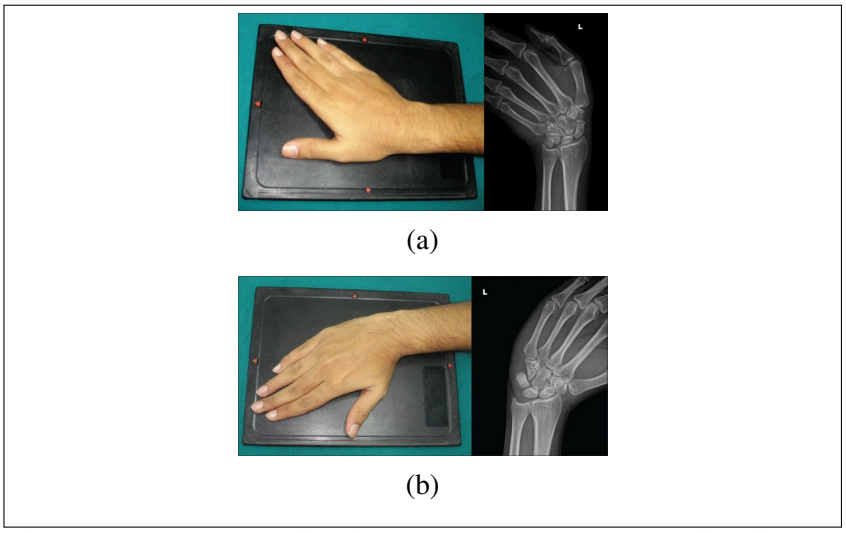


Figure 3.4. Oblique view X-ray imaging [61].

On the other hand, in a scaphoid view, the wrist is placed in ulnar deviation as the hand is tilted toward the pinky side which can be observed in Figure 3.5a. The X-ray beam is directed dorsally, making this angle a subset of the posteroanterior view. It is often utilized to maximize the visualization of the scaphoid bone, which may be covered in standard views. In the carpal tunnel view, the wrist is extended dorsally, and the X-ray beam is angled to visualize the volar aspect of the wrist such that carpal tunnel and adjacent structures are underlined . Last but not least, in the radial deviation view, the wrist is tilted toward the thumb side as opposed to the ulnar deviation, placing stress on the ulnar structures which makes it easy to evaluate alignment and ligament stability. Figure 3.5b illustrates an example radial deviation view.

Despite the fact that X-ray images are the gold standard for the detection of DRFs, diverse imaging modalities are explored in the literature, mostly to supplement radiographs. One of the alternative imaging techniques is ultrasounds. Ultrasound imaging, also known as sonography, utilizes high-frequency sound waves to construct images of the inside of the body [64]. A transducer, the part of the ultrasound machine held in hand by a clinician is placed on the skin and moved over the area of interest, emits sound waves of between 2 and 15 Mega Hertz (MHz) into the body. The sound travels through the tissues and are reflected back when they confront a boundary. The reflected sound waves that differ based on the density and



(a)

(b)

Figure 3.5. (a) Scaphoid angle, (b) Radial deviation angle [61].

composition of the tissues are detected by the transducer and these waves are processed by the ultrasound machine so that a real-time image is displayed on the ultrasound monitor. Similar to X-ray, bone reflects the most sound waves and appears white on the images while fluids transmit sound waves, appearing darker. This technique is utilized commonly for exploring non-radiative diagnosis, especially for pediatric patients, and repeated use. Furthermore, ultrasound machines are portable, making them suitable for bedside point-of-care and for diagnosis in emergency scenarios.

Computed tomography (CT), on the other hand, uses radiation as it utilizes X-rays [65]. Advanced computer processing algorithms generate 3D reconstructions to deliver detailed cross-sectional images of the body. Compared to standard radiographs, CT creates a more detailed view of bones and tissues. In the literature, CT is commonly employed to detect subtle fractures that are not visible on radiographs or in cases that necessitate precise anatomical mapping for evaluating bone fragment displacement.

3.2. IMAGE PROCESSING AND ANNOTATION

Annotation is the procedure of labeling data to deliver ground truth information for the model to learn from and it is another crucial process for the preparation of the dataset. In medical imaging-related computer vision tasks, expert clinicians such as experienced radiologists or

surgeons are required to manually mark the area of interest, delivering ground truth or gold standard. Fracture detection annotations generally consist of marking the fracture regions with either bounding boxes or segmentation masks. An alternative to this is classifying the images as fractured or non-fractured.

There are various ways to achieve annotations. Most commonly, bounding boxes are utilized to highlight the ROI where the fractures are located by enclosing the fracture area in a rectangular box. As shown in Figure 3.6, the red bounding box marks the fracture in the radiograph and labels it as ‘‘fracture’’. The coordinates and labels of the bounding box are saved in a text-like file for training machine learning models. During the training phase, the model pairs the image data with the corresponding ground truth information. While different models require the data on the annotation file to be formatted in different ways, the overall principle remains the same.

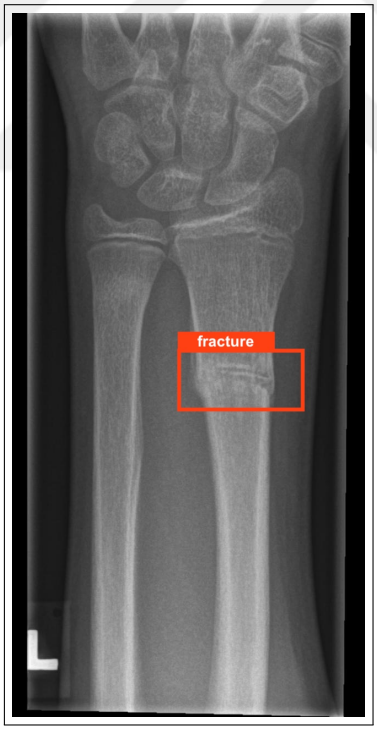


Figure 3.6. Bounding box annotation of a fracture.

Bounding boxes are defined by their coordinates. Coordinates can be absolute or relative. In absolute coordinate system, the top-left and bottom-right corners of the bounding box and its label are present, named ‘‘xmin’’, ‘‘ymin’’, ‘‘xmax’’, and ‘‘ymax’’, as shown in Figure 3.7. For example, 50, 75, 200, 300 means that the box starts at (50, 75) and extends to (200, 300).

In relative coordinate system, the bounding box's coordinates are given relative to the image size, named ‘‘center_x’’, ‘‘center_y’’, ‘‘width’’, and ‘‘height’’. For example, if an image has a resolution of 1024x1024 and the box is centered at (512, 512) with a width and height of 256, the relative values would be 0.5, 0.5, 0.25, and 0.25.



Figure 3.7. Annotation of a bounding box.

During the training phase, the model takes the annotation files and images as inputs. Typically, there is a preprocessing step that converts the annotation files into tensors or arrays, enabling the model to process the information. This processing procedure can vary from one model to another. The model compares its predictions to the ground truth from the annotations in order to calculate loss functions such as Intersection over Union (IoU) or binary cross entropy to reveal the error for bounding box regression. Furthermore, most models continuously adjust their weights to minimize the difference between predicted and true bounding boxes during the training.

In binary labeling, a ‘‘yes’’ or ‘‘no’’ label is assigned to the entire image. Models that learn from these types of annotation only perform classification rather than detection and localization. An example of this approach would be to mark an entire X-ray to be ”fractured” or ”not-fractured”. The segmentation mask annotation goal, on the other hand, is to delineate the area of interest at a pixel level. In this type of annotation, each pixel is assigned to a class. It is particularly useful for grayscale or color images where different pixel values indicate different objects. There are three types of segmentation, namely semantic, instance, and panoptic. In semantic segmentation, every pixel in the image is labeled with a class. Pixels of the same class share the same label, regardless of object instances. In instance segmentation, however, there is a distinguishment between different instances of the same class. Lastly, in panoptic segmentation, semantic and instance segmentation are combined, thus, all pixels are both labeled and distinguished between the instances of the same object.

The annotation process can be manual, semi-automated or fully automated. While manual

annotation uses tools like polygons, brushes, or freehand drawing such that the annotators manually label the ROI, semi-automated annotation utilizes tools that may be AI-assisted to speed up the process as they can suggest regions or boundaries which are refined by human annotators later on. Lastly, fully automated techniques use ML models to generate masks. However, this approach often requires human correction or verification. Labelbox [66], Roboflow [67], computer vision annotation tool (CVAT) [68], and RectLabel [69] are some of the tools for the annotation process.

While bounding boxes are adequate for object detection tasks, they provide only a rough approximation of the ROI as they enclose the target within a rectangular area. However, for instance segmentation tasks, where the goal is to predict object masks at a pixel level, bounding boxes are inappropriate as they include irrelevant background pixels. Instead, polygon annotations are utilized to outline the fracture region with precision.

Polygon annotation involves marking the exact contour of the ROI by defining a set of coordinate points that trace its boundaries. Unlike bounding boxes, which store only two corner coordinates, polygon annotations store multiple (x, y) coordinate pairs, enabling a more detailed and pixel-accurate representation of the object.

In order to train instance segmentation models, polygon annotations are necessary since they enable the generation of pixel-wise masks rather than simple bounding boxes. During training, the model learns to predict these fine-grained masks, allowing for improved localization and contour accuracy of detected fractures. Figure 3.8 presents a sample polygon annotation executed on the fractures from the dataset. This pixel-based approach enhances the model's ability to differentiate overlapping instances and better understand the morphological characteristics of fractures [70].

Data splitting is the next step in training machine learning models and is of paramount importance in computer vision tasks, as proper splitting ensures that the model generalizes well to unseen data while avoiding underfitting or overfitting. Underfitting occurs when a model is too simple to capture the underlying structure of the data, leading to poor performance on both training and testing sets. Overfitting, on the other hand, happens when a model learns the training data too well—including noise or irrelevant details and fails to generalize to new data. The model might be too complex for a given task, the training set

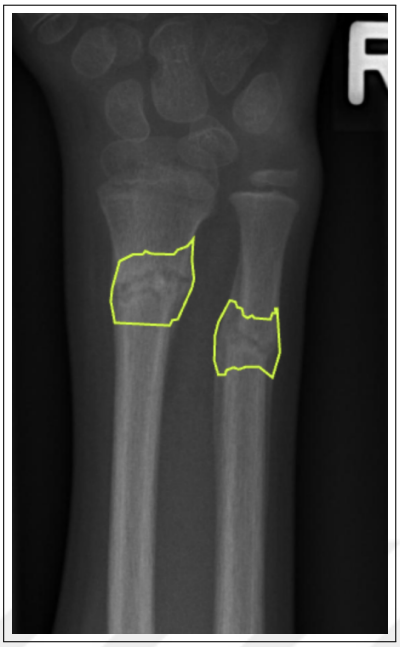


Figure 3.8. Polygon annotations on fractures.

may be too small or not representative, or the model is trained for too many epochs, which causes the model to memorize the training data.

There are three types of data splits, namely training, validation, and testing sets. The largest portion of the dataset is allocated to the training set, as it is utilized to train the model. Approximately 60-80% of the data is rationed to the training set typically. In comparison, the validation set is employed by the model to tune its hyperparameters during the training, with the objective of avoiding overfitting. Generally, 10% of the overall data is assigned to this set, however, a model does not strictly require a validation set to conduct training. Regardless, using a validation set is beneficial as it allows the tuning of hyperparameters and early halting of the training to prevent overfitting. Furthermore, it provides a way to monitor the model's generalization performance during the training. Lastly, the testing set is utilized to evaluate the final model's performance on unseen data once the training is complete. 10% of data is reserved for testing purposes commonly, however, additional data can also be utilized.

One of the key considerations for data splitting is randomization. Data in the dataset should be shuffled prior to splitting to avoid bias. Bias is the error or unfairness in the model's predictions. It can occur due to the dataset, model architecture, or the deductions made by the model during the training process. Consequently, it leads to skewed or inaccurate predictions,

especially for complex or underrepresented classes. A model with a high bias makes strong assumptions about the data and this happens because the model is too simple to grasp the underlying patterns behind its predictions, which often leads to underfitting. This can be due to the model's simplicity, it may not have enough parameters or layers, the model could be trained for too few epochs, the features highlighted in the annotation phase may not be informative enough for the model to learn or there may be too few instances in the dataset. Another crucial concept is the bias-variance tradeoff. Variance is the model's sensitivity to small changes in the training data such that a model should find the proper balance between bias and variance to minimize the total error. Total error is presented in Equation (3.1) where the irreducible error stems from artefacts ingrained in the data that no model can eliminate.

$$\text{TotalError} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error} \tag{3.1}$$

A well-performing model should have a low bias and no overfitting or underfitting, it should make the least amount of assumptions possible by learning the complex patterns present in the training data while staying flexible. Equation (3.2) presents the calculation of bias for model f(x) by measuring the difference between the expected or average prediction of a model and the ground truth. $\mathbb{E}[f(x)]$ is the expected value of the model while y is the true target value.

$$\text{Bias} = \mathbb{E}\left[f(x)\right] - y \tag{3.2}$$

Especially for imbalanced datasets, stratification is equally as important as randomization during data splitting. Stratification refers to maintaining the same class distribution in training, validation, and testing sets. For sequential data such as video frames, temporal splitting may be applied to evade data leakage. Data leakage should be avoided by ensuring there is no overlap between training, validation, and testing sets. This problem typically arises from using augmented versions of the same image in different sets.

After data splitting, the data should be prepared to train and infer computer vision models by enhancing consistency and features while reducing artifacts. Different datasets, model types, and tasks require different preprocessing approaches, however, resizing is one of the most

common preprocessing steps. This procedure ensures that all images have a fixed dimension which reduces computational cost and memory usage for training. It should be used when input images have inconsistent resolutions and the model requires a fixed input size. However, it should be avoided if the aspect ratio is critical such that resizing distorts the image. For resizing, interpolation is a commonly used mathematical process to estimate unknown values between known data points [71,72,73]. Likewise, it is utilized for resizing, rotating, and transforming images by calculating pixel values that are not present in the initial image. One of the most common interpolation methods is the nearest neighbor which works by assigning the value of the nearest known pixel to the new pixel. Equation (3.3) shows the formula for the nearest neighbor interpolation where *(sx, sy)* are the scaled coordinates of the new pixel. While it is fast and does not induce computational complexity, it can provide blocky images with aliasing, making it unsuitable for smooth enlargements.

$$I'(x, y) = I(\text{round}(sx), \text{round}(sy)) \tag{3.3}$$

Bilinear interpolation, on the other hand, is better for achieving smoother images. It employs the weighted average of the four closest pixel values, as presented in Equation (3.4) where $(x_1, y_1), (x_2, y_1), (x_1, y_2), (x_2, y_2)$ are the four nearest pixels to the original pixels and dx and dy are the distances from the new pixel to the original pixel. While it produces smoother results compared to the nearest neighbor and is computationally not complex, it comes with a slight blurring effect.

$$\begin{aligned} I'(x, y) =& (1 - dx)(1 - dy)I(x_1, y_1) + dx(1 - dy)I(x_2, y_1) \\ &+ (1 - dx)dyI(x_1, y_2) + dxdyI(x_2, y_2) \end{aligned} \tag{3.4}$$

Normalization is another type of image-preprocessing technique that scales the image's pixel values to a range that improves model convergence and stability. It should be implemented if the model training is unstable because of pixel value variations or while training with pre-trained models that require particular input distributions. However, it should be avoided while working with image compression formats that already normalize values or when raw

pixel intensities are directly meaningful, like that of X-ray images. One type of normalization is the min-max normalization, shown in Equation (3.5). It works by scaling pixel values as [0, 1] or [-1, 1]. Z-Score normalization, on the other hand, is shown in Equation (3.6) where μ is the mean and σ is the standard deviation. The pixel values are transformed such that they have a mean of 0 and a standard deviation of 1, resulting in pixel intensities within a scale that enables the models to train more efficiently. Lastly, mean subtraction is a simpler normalization alternative where the mean pixel value of either the dataset or an image is subtracted from each pixel. It should be employed when data is required to be centered around zero without changing the scale and using models that expect zero-centered input. The formula is presented in Equation (3.7) where $I(x, y)$ is the original pixel intensity, μ is the mean intensity of the dataset or an image, and $I'(x, y)$ is the altered pixel value.

$$I'(x, y) = \frac{I(x, y) - \min(I)}{\max(I) - \min(I)} \tag{3.5}$$

$$I'(x, y) = \frac{I(x, y) - \mu}{\sigma} \tag{3.6}$$

$$I'(x, y) = I(x, y) - \mu \tag{3.7}$$

Histogram equalization, on the other hand, improves the contrast of an image by redistributing pixel intensity values. The formula is presented in Equation (3.8) where CDF is the cumulative distribution function and L is the number of intensity levels. It should be used while improving visibility in low-contrast images or while dealing with poor lighting conditions. Nonetheless, it should not be employed when image intensity variations add meaning to the images or while working with color images.

$$I'(x, y) = \frac{\text{CDF}(I(x, y)) - \min(\text{CDF})}{\max(\text{CDF}) - \min(\text{CDF})} \times (L - 1) \tag{3.8}$$

Denoising is the process of removing noise from the images in the dataset without affecting important details [74]. Noise can be due to low light conditions, bad image quality, or imaging

artifacts. One of the most common approaches for denoising is Gaussian smoothing, which is achieved by removing sensor noise caused by low-light circumstances based on the Gaussian kernel. In Equation (3.9), the Gaussian distribution in one dimension is shown where σ is the standard deviation of the distribution. Furthermore, Figure 3.9 shows the same image before and after Gaussian smoothing.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.9}$$

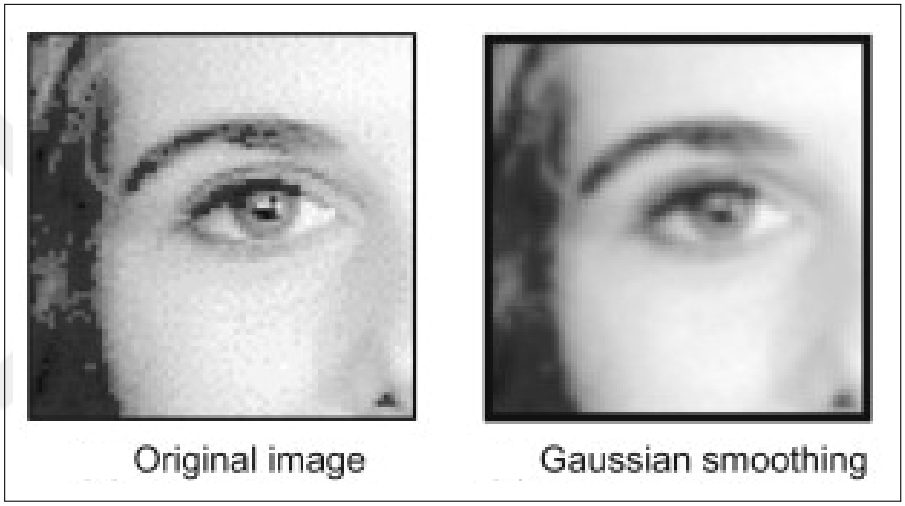


Figure 3.9. Gaussian operation on the same image [75].

Last but not least, color space transformations convert images into different color ranges to enhance feature extraction from the dataset. One of the most common color transformations is the grayscale conversion which converts a colored image to grayscale. Another alternative conversion to Hue, Saturation, Value color space, which is a cylindrical representation of colors such that it has a closer alignment compared to the red-green-blue (RGB) model in terms of human perception of colors. Figure 3.10 shows the color spaces for RGB and HSV models.

Image augmentation is a procedure commonly utilized in computer vision tasks. It expands the dataset size and diversity by applying various transformations to the existing images. Contrarily to humans, neural networks do not ‘‘see’’ the images. Rather, they process the input images as matrices of pixel values, therefore, by applying image augmentation methods, new variations of the same image are added to the dataset, which is recognized as an entirely

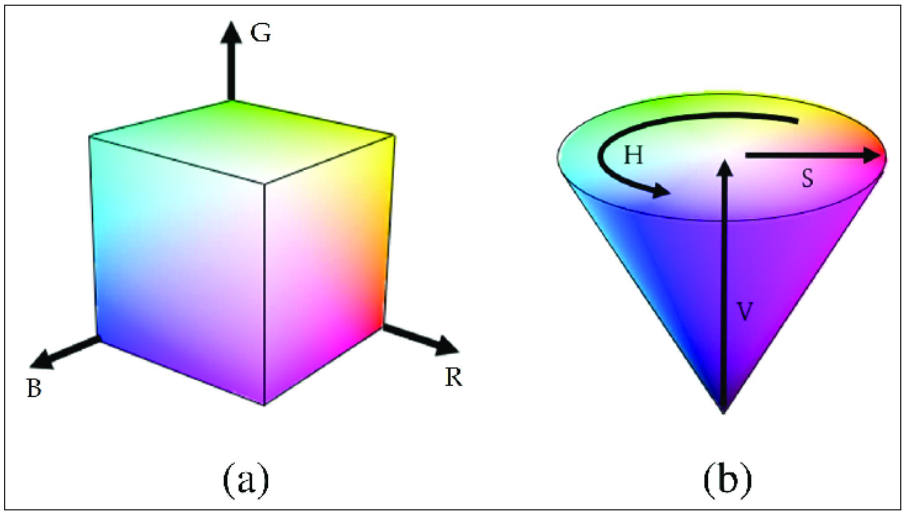


Figure 3.10. RGB versus HSV color space [76].

new image from the perspective of the model. Models typically learn patterns by detecting edges, textures, colors, and shapes in the data at different spatial locations. Thus, even minor modifications performed on an image alter these patterns such that the model can learn generalized features instead of memorizing specific pixel placements, which may lead to overfitting. Equation (3.10) shows the increasing of a dataset if it initially consists of N images, and k unique augmentations are applied to each image. For example, if the dataset consists of 1000 images initially and 5 augmentation steps are applied to each image, the final dataset would be expanded to 5000 images.

$$N_{\text{augmented}} = N \times k \tag{3.10}$$

Types of augmentation can be classified into 4 different categories, namely, geometric, color-based, noise-based, and advanced transformations. Geometric transformations consist of altering the spatial properties of an image. For instance, the image can be rotated by a specific degree, which helps the model recognize objects regardless of their orientation. Another alternative is flipping the image horizontally or vertically, which mirrors the image either along the vertical axis or the horizontal axis. Cropping, on the other hand, extracts a region from the original image and helps the model to focus on object-specific regions. Figure 3.11 shows the same image before and after the rotation and flipping augmentation procedures.

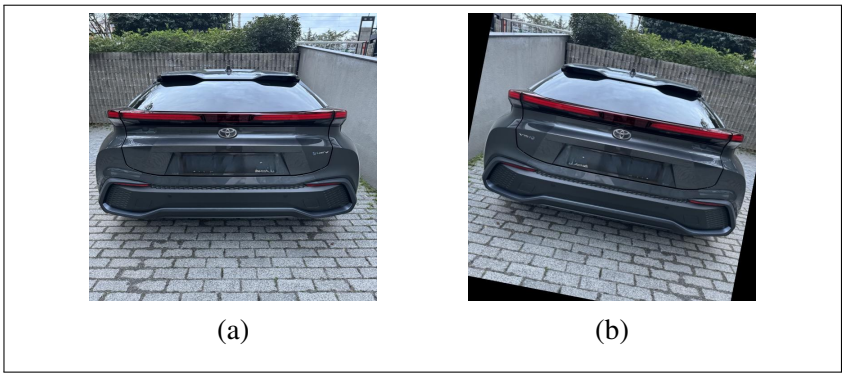


Figure 3.11. (a) Original image, (b) Image after rotation and flipping augmentations.

Color-based transformations, on the other hand, work by modifying the values for pixel intensity. Brightness, contrast adjustments, and grayscale conversions can be listed under this heading. While brightness adjusts the illumination by scaling pixel intensities, contrast augmentation alters the pixel intensity values by stretching or compressing them. Figure 3.12 presents the image after contrast and brightness augmentations.



Figure 3.12. Image after brightness and contrast augmentations.

Noise-based transformations introduce random variations to the images to simulate real-world artifacts, making the model's predictions more immune to such cases. Salt and pepper noise randomly turns some pixels black and white while Gaussian noise adds random noise to simulate sensor imperfections. Lastly, advanced augmentation methods include the cutout (random erasing), mixup, and generative augmentations. Cutout randomly removes parts of an image to push the model to learn robust features. Mixup combines two or more images and their labels into a single image. A more novel solution is using Generative Adversarial Networks (GANs) to create new, realistic, and artificial training images.

3.3. MODEL ARCHITECTURES

This section describes various algorithms for DRF detection, localization and segmentation in detail. Essential information for commonly utilized frameworks and their implementation are discussed. Furthermore, additional models for ensuring model interpretability are also explained, as the interpretability of a model is of utmost importance for clinical adoption.

Convolutional Neural Network (CNN) is a type of deep learning architecture that processes grid-like structured data, such as images [77,78]. Unlike the traditional fully connected neural networks, they leverage the spatial relationships between pixels as they use convolutional layers to learn patterns present in images, such as edges, textures, and shapes. Traditional neural networks, on the other hand, treat each pixel of an image independently and do not exploit spatial locality like CNNs, which expands the required number of parameters significantly. CNNs are especially influential for tasks like image classification, object detection, and segmentation since they have the ability to learn spatial hierarchies of features from the input images automatically and adaptively. Figure 3.13 depicts the detailed structure and components of a CNN, consisting of convolutional layers, activation functions, pooling layers, fully connected layers, loss functions, and forward/back propagation.

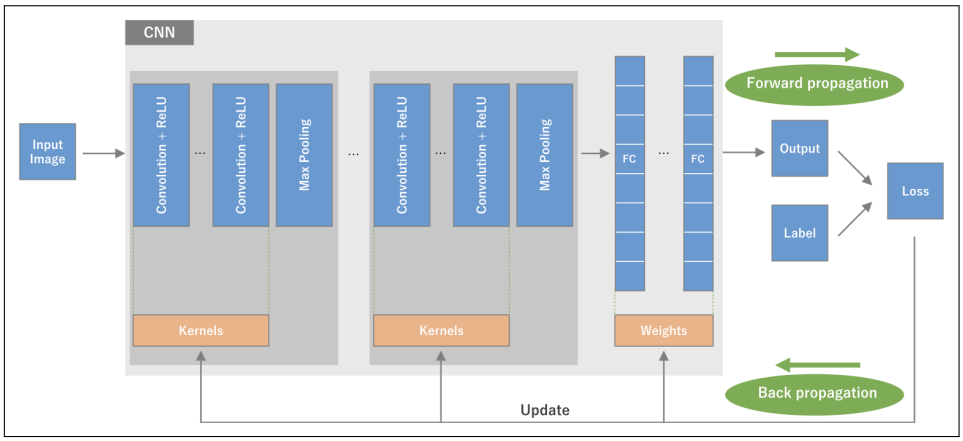


Figure 3.13. CNN architecture structure [78].

Convolutional layers are the building blocks of a CNN structure as they perform the convolution operations on the input images. These convolution operations extract different features such as edges, textures, and patterns from the input image by using a kernel over the image to produce a feature map. A kernel, also known as a filter, is a small matrix of

learnable weights in convolutional layers to detect patterns from a given image. It produces the feature map by sliding over the image. Typically, they are of size 3x3 or 5x5 without padding. Figure 3.14 shows a convolution operation with a 3x3 filter size on an input tensor, performing an element-wise multiplication followed by summation, creating a feature map consequently.

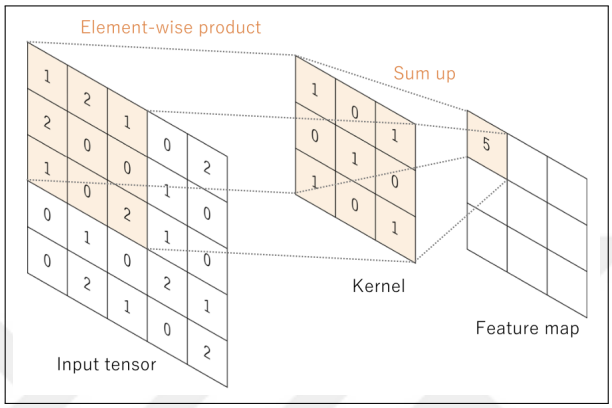


Figure 3.14. Convolution operation using 3x3 kernel [78].

The map is then passed on to the subsequent layer. The convolution operation is shown in Equation (3.11) mathematically where I is the input image, K is the kernel or the filter, *(i,j)* are the spatial coordinates in the output feature map.

$$(I * K)(i, j) = \sum_{m} \sum_{n} I(i - m, j - n) \cdot K(m, n) \tag{3.11}$$

The critical hyperparameters for the convolution layer can be listed as kernel size, stride, and padding. They influence the performance of the CNN in terms of output feature maps, information retention, and computational complexity. While a small-sized kernel can capture refined details, a larger kernel can capture more general patterns, overlooking finer details. Furthermore, smaller kernels reduce the number of parameters, resulting in more efficient training. However, stacking multiple kernels might enable better feature extraction with fewer parameters. Stride, on the other hand, is the step size taken while moving the filter across the input image. If the stride is equal to 1, the filter moves pixel-by-pixel on the image while if it is equal to 2, it skips every other pixel. Smaller stride retains more spatial details and produces a larger feature map, however, has a higher computational cost. A larger stride reduces the computational cost as it downsamples the output map faster, yet, it may be insensitive

to finer details. The output feature map size can be calculated with the given formula in Equation (3.12) where O is the map size, I is the input size, k is the kernel size, and s is the stride value.

$$O = \frac{I - k}{s} + 1 \tag{3.12}$$

Lastly, padding is the technique of adding extra pixels around the input image, which are commonly zeros, with the goal of controlling the output size. Padding can be set to zero such that the feature map shrinks as the convolution operation progresses, which results in less context for edge pixels and a loss in border information. If zero padding is applied, which means that the padding parameter is equal to a value larger than 0, the output size is maintained to be the same as the input size. This helps preserve spatial features at edges while enabling deeper networks with more layers without excessive shrinking. For instance, Figure 3.15 illustrates an input tensor where zero padding is applied by marking the padded parts around the tensor with dashed lines.

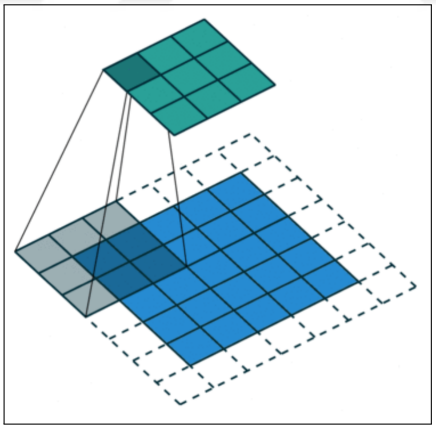


Figure 3.15. Convolution operation on a zero padded input tensor [77].

Following the convolution layer, an activation function is applied to the network to learn complex patterns by presenting non-linearity. These functions take the output of a neuron and convert it non-linearly to decide whether a neuron should be activated or not. Commonly applied activation functions are Rectified Linear Unit (ReLU), Sigmoid, and hyperbolic tangent (Tanh) functions. The graphs for the functions can be observed in Figure 3.16.

Activation functions are conducted after each convolution operation in a CNN as well as

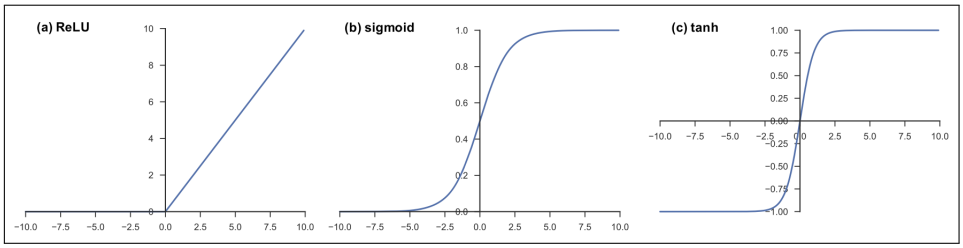


Figure 3.16. Common activation function graphs [78].

in fully connected layers before the final classification. They help prevent vanishing or exploding gradients while improving training stability by ensuring efficient weight updates. Different activation functions have distinctive influences on CNN behavior. $ReLU$, shown in Equation (3.13), solves the vanishing gradient problem, which occurs due to gradients that update the network becoming so small that they vanish while they are backpropagated from the output layers to the initial layers. It also provides faster training as it enables efficient gradient propagation, simplicity, and computational efficiency. However, it may present with a different problem such that all of the neurons get stuck at unactivated, which is known as the dying ReLU problem. The *Sigmoid* activation function, presented in Equation (3.14), on the other hand, outputs a value between 0 and 1 for neurons. However, this results in the vanishing gradient problem as large inputs shrink gradients to zero. Furthermore, it is not suitable for deep CNNs with a lot of layers and depth, as it yields slow learning. Similar to $Sigmoid$, $Tanh$, presented in Equation (3.15), also suffers from vanishing gradients as it outputs a value between -1 and 1. Nonetheless, it performs better zero-centered activation compared to *Sigmoid* and can reduce bias shifts during training.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3.13}$$

$$\mathrm{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3.14}$$

$$\mathrm{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.15}$$

The pooling layer follows the activation functions and they reduce the spatial dimensions of the feature maps, which makes the model computationally more efficient and robust to spatial variations while controlling overfitting. It is essentially a downsampling operation as it reduces the height and width of feature maps while keeping the important features

and discarding irrelevant information. Furthermore, it provides translation invariance such that small shifts in the input image does not change the outputted future maps. The two most commonly used pooling operations are max and average pooling. Max pooling selects the maximum value in a defined window of the feature map with the formula shown in Equation (3.16) where $X_{\mathrm{m,\,n}}$ are the pixel values in the receptive field. It keeps the strongest features while reducing sensitivity to noise and preventing overfitting by forcing abstraction. Average pooling, on the other hand, takes the average of the values defined in a certain window using the formula shown in Equation (3.17) where Xm, n are the values inside the pooling window. It preserves the general spatial structure and helps smooth feature maps. However, it should be utilized in low-detail requiring tasks such as segmentation.

$$S(i,j) = \max(X_{m,n}), \quad m,n \in \text{window} \tag{3.16}$$

$$S(i,j) = \frac{1}{m \times n} \sum X_{m,n} \tag{3.17}$$

Hyperparameters for the pooling layer are the same as the convolutional layer where filter size defines the receptive field for pooling, stride refers to the step size of the pooling window, and padding affects the output size. Figure 3.17 shows the comparison of the max and average pooling procedure of a 4x4 input tensor with a filter size of 2x2 and stride of 2 side by side.

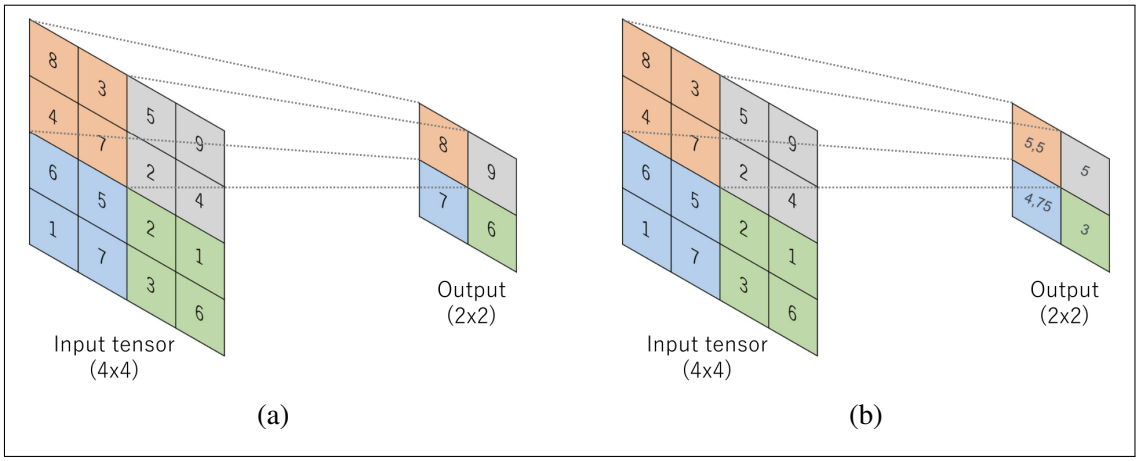


Figure 3.17. (a) Max pooling [78], (b) average pooling.

Subsequently, after multiple convolutional and pooling layers, the extracted features are flattened into a 1-dimensional vector and it is delivered to the fully connected (FC) layer to perform high-level reasoning in the neural network that performs classification by utilizing the

softmax function shown in Equation (3.18) for calculating class probabilities. In the formula, z_i is the input to the softmax function for class i and the resulting value is the class probability for that particular class. Every single neuron in an FC layer is connected to every neuron in the previous layer.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3.18}$$

Once the convolutional, activation, pooling, and fully connected layers have been structured, the next phase is to train the CNN to optimize its parameters and enhance its performance on the given task. Training a CNN involves adjusting the weights of the filters and neurons in such a way that the network accurately predicts the output based on the input image. This is achieved by minimizing the difference between the CNN's predictions and the actual ground truth labels using a loss function, which is an optimization process that iteratively updates the weights through backpropagation and gradient descent algorithms. It quantifies the difference between the predicted and actual outputs of the CNN. While training, the aim of the neural network is to minimize the value returned by the loss function. One of the most commonly employed loss functions is the cross-entropy loss function, as it works especially well for tasks where the model needs to classify multiple classes and heavily penalizes incorrect predictions. The formula for the cross-entropy loss function is shown in Equation (3.19) where y_i is the true label, which is valued at 1 if the class prediction is true and 0 otherwise, $_i$ refers to the predicted probability for class i.

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3.19}$$

The other commonly used loss function is the Mean Squared Error (MSE) function, which is utilized for regression tasks where the model needs to perform continuous value predictions. However, it is not ideal for classification tasks, since it is highly sensitive to outliers. After loss computation, optimization algorithms are utilized to update the weights of the CNN to reduce the calculated loss values. Gradient Descent (GD) is one of the well-known algorithms that update the weights of the CNN by moving in the direction of the steepest decline of the loss function. The update rule of the GD algorithm is shown in Equation (3.20), where W is

the weight parameter, α is the learning rate, and $\frac{\partial L}{\partial W}$ is the gradient of the loss function with respect to weight W.

$$W = W - \alpha \frac{\partial L}{\partial W} \tag{3.20}$$

Furthermore, Figure 3.18 illustrates the GD optimization algorithm. The blue line shows the loss function curve that represents the loss function CNN is attempting to minimize and the orange lines show the slope or gradient of the loss function at different points to determine the direction of weight updates. Blue arrows, on the other hand, show the steps taken downward which represent the learning process. Each step, whose size is defined by α, moves in the direction of the steepest descent to minimize the loss. The α value has a crucial effect on the optimization process. If it is set to be too large, the steps might overshoot the global minimum value. Contrarily, if it is too small, the convergence to the global minimum error value will be slow.

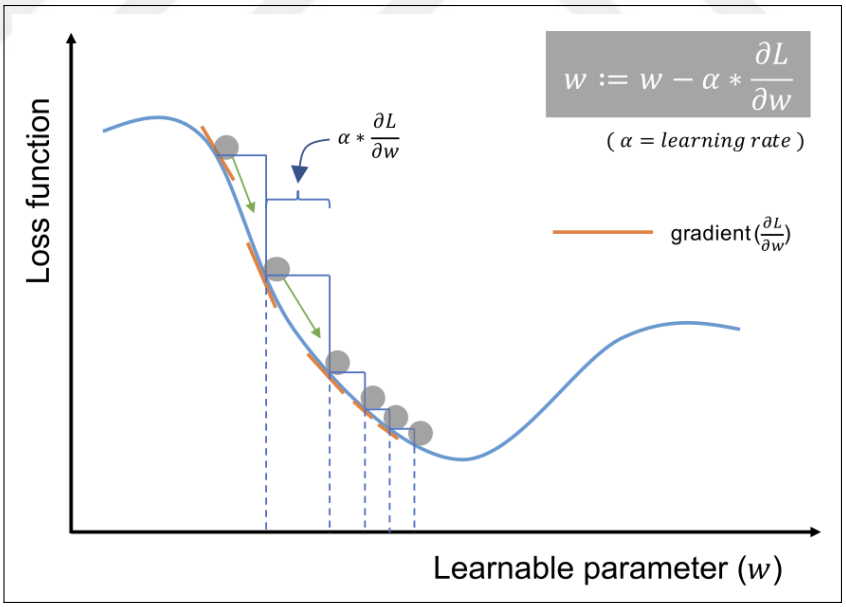


Figure 3.18. GD optimization algorithm process [78].

There are different types of approaches for GD optimization. While Batch GD uses the entire dataset to compute gradients, which yields more accurate updates, it is slower and more intense on memory. Mini-batch GD, on the other hand, uses small batches from the dataset rather than the whole dataset, achieving a balance between speed and accuracy. Lastly,

stochastic GD updates the weights using one sample at a time, performing faster yet noisier updates.

Another commonly used optimization algorithm is the Adaptive Moment Estimation (Adam Optimizer). It achieves efficient and fast convergence by combining the advantages of Momentum-based GD and Root Mean Square propagation. It can adjust its learning rates per parameter, providing adaptive learning rates, thus performing well for large datasets and noisy gradients.

CNNs need to decide how much each weight contributes to the overall loss to productively update their weights. Backpropagation is the algorithm that calculates the gradient of the loss function with respect to each parameter in the network and propagates the error backward from the output layer to the initial, shallow layers. This ensures that each filter and neuron is adjusted properly based on its role in the final prediction by utilizing the chain rule of differentiation to adequately compute the gradients and allowing the optimization algorithms to update their weights accordingly. It consists of three main phases, namely forward pass, loss computation, and backward pass.

Forward pass performs prediction computation. During this stage, the CNN conducts convolution, activation, and fully connected layers to perform predictions on the input image. Thereafter, the chosen loss function measures the error between the predicted output and the ground truth. In the backward pass phase, backpropagation calculates gradients of the loss function with respect to each weight present in the CNN and once they are computed, weights are updated using GD. This process is repeated iteratively until the model reaches convergence.

Residual Networks (ResNet) is a deep learning architecture designed to overcome the vanishing gradient problem while training deeper neural networks [79]. Consequently, it is designed to allow the training of deep networks with up to 152 layers while maintaining accuracy and efficiency by utilizing residual learning. Residual learning employs skip connections, also known as shortcut connections, to improve the flow of gradient and mitigating optimization. While constructing a deep CNN architecture improves feature learning, increasing the depth beyond a certain point often leads to higher training errors rather than improved model performance. This degradation may happen even when there

is no overfitting for traditional CNNs. Although deeper CNNs are theoretically expected to yield higher accuracy, in practice, they often perform worse due to vanishing gradients and optimization difficulties. This phenomenon is shown in Figure 3.19, where the training error for a 56-layer plain network is significantly higher compared to a 20-layer plain network.

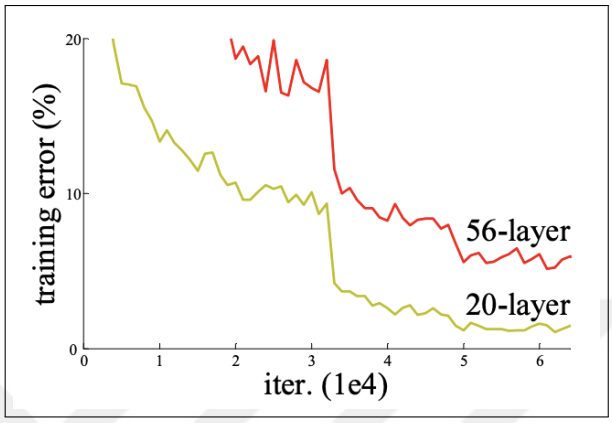


Figure 3.19. Training error comparison of a shallow versus deep network [79].

ResNet architecture solves this problem by utilizing residual learning and utilizes a residual block equation shown in Equation (3.22), instead of learning a direct mapping, presented in Equation (3.21). In the equations, x is the input, $H(x)$ is the desired function, and $F(x)$ is the residual function, which is the difference between $H(x)$ and x. This means that ResNet learns the extent of the required adjustment, rather than learning the entire mapping from scratch.

$$H(x) = F(x) + x \tag{3.21}$$

$$F(x) = H(x) - x \tag{3.22}$$

Residual learning bypasses certain layers using shortcut connections. These connections relieve the burden of learning complex transformations, allow gradients to flow directly through the network during backpropagation, and prevent problems related to deep networks. Figure 3.20 shows a plain 18-layer CNN along with an 18-layer ResNet architecture where the skip connections are shown with black downward arrows. They bypass one or more layers in a deep neural network, which allows the original input to be added directly to the output of a later layer. They are crucial for ResNet architecture's ability to avoid the vanishing gradient problem.

The ResNet architecture consists of multiple residual blocks stacked together where each

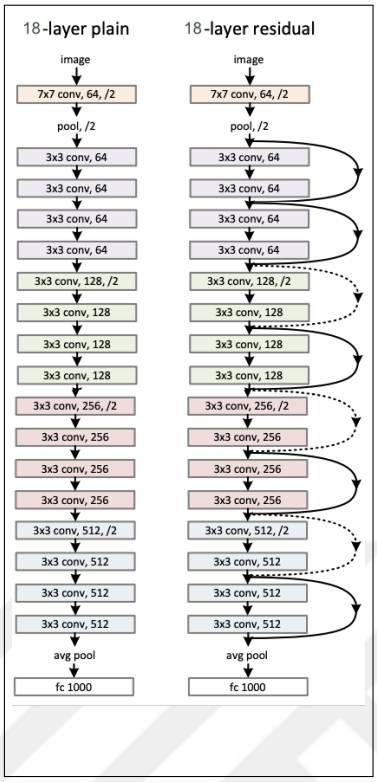


Figure 3.20. Skip connections in a 18-layer ResNet [79].

block contains convolutional layers, batch normalication, ReLU activation, and shortcut connections. There are different variants of ResNet, namely ResNet-18 with 18 layers, ResNet-34 with 34 layers, ResNet-50 with 50 layers, ResNet-101 with 101 layers, and ResNet-152 with 152 layers. These variants differ in terms of the number of parameters and floating point operations per second (FLOPS). Deeper layered architectures have more parameters and FLOPS, however, they have a mechanism to reduce computation known as bottleneck blocks. ResNet-50 and beyond utilize bottleneck blocks, which use 1x1 convolution to perform channel reduction, 3x3 convolution to perform computation, and 1x1 convolution to expand the channels back instead of using two 3x3 convolutions. Figure 3.21 shows the implementation of the bottleneck structure on the right-hand side, instead of the two 3x3 convolutional layers shown on the left-hand side.

EfficientNet is another type of CNN framework that introduces a novel compound scaling method, which uniformly combines depth, width, and resolution scaling[80]. Depth scaling increases the layers in the network architecture, making it deeper. Width scaling, on the other hand, increases the channels present in the architecture, creating a wider network. Lastly,

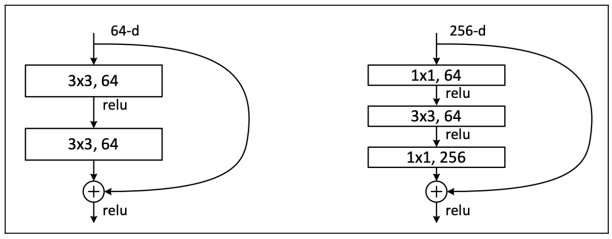


Figure 3.21. Illustration for a bottleneck block [79].

resolution scaling increases the input image size, which improves fine-grained detail learning in return. The compound scaling EfficientNet provides the balancing of these dimensions optimally, while traditional CNNs arbitrarily increase said dimensions. Traditional scaling leads to a decrease in accuracy, inefficient usage of resources, and substandard model performance as it scales only one dimension. EfficientNet, however, solves this problem by scaling all three dimensions uniformly with a compound coefficient. Furthermore, it gets rid of the requirement of manual hyperparameter tuning. Figure 3.22 illustrates the compound scaling such that (a) shows a standard CNN structure, (b) presents a wider CNN with width scaling, (c) displays a deeper CNN using depth scaling, (d) depicts a higher resolution CNN with resolution scaling, and (e) shows a CNN with all scales applied using EfficientNet.

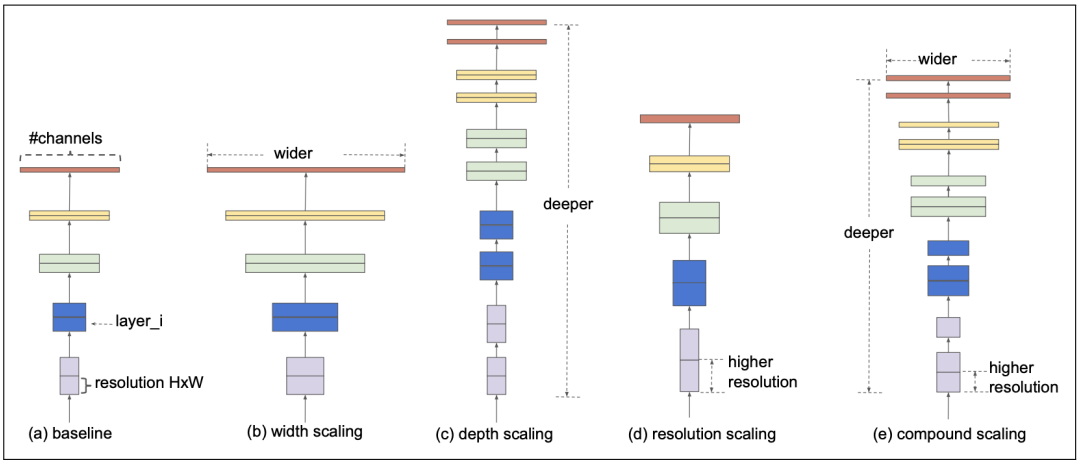


Figure 3.22. CNN scaling [80].

EfficientNet's backbone is based on Mobile Network version 2. MobileNet is a lightweight deep learning architecture that is optimized for mobile and embedded vision applications and Mobile Network version 2 (MobileNetV2) improves upon V1 by introducing inverted residual blocks and linear bottlenecks. MobileNetV2 has a reduced model size, computational cost, and latency compared to ResNet and VGG by employing depthwise separable convolutions

rather than traditional convolution layers. This process applies a single filter per input channel while pointwise convolution (1x1 convolution) combines the outputs from the depthwise convolutions, which results in approximately 9 times fewer computations compared to standard convolutions, with only a minor loss in accuracy. Furthermore, it utilizes inverted residual connections, which is a functionality derived from ResNet. However, unlike ResNet residual blocks that expand the feature space, MobileNetV2 first compresses it, performs computations, and then expands it back. This approach reduces memory usage and computational cost while allowing direct gradient flow for stable training.

However, while MobileNetV2 focuses on efficiency, it does not incorporate an optimal scaling strategy for balancing depth, width, and resolution across different layers of the network as EfficientNet does. EfficientNet achieves state-of-the-art performance while maintaining significantly lower computational costs than traditional CNN architectures like ResNet and DenseNet by utilizing MobileNetV2 as its backbone.

Furthermore, EfficientNet integrates Squeeze-and-Excitation (SE) blocks into its architectures to improve feature representation and optimization. SE blocks have 3 essential components, namely the squeeze step, excitation step, and feature recalibration. The squeeze step involves applying global average pooling which extracts a single value per feature map. The excitation step, on the other hand, consists of a small fully connected network that learns which features to emphasize. Based on this process, feature recalibration gives increased attention to important features while suppressing irrelevant features. Consequently, SE blocks improve model accuracy without too much additional computation cost and enhance important features while reducing noise.

EfficientNet has 8 versions, ranging from B0 to B7, each increasing in complexity, accuracy, and computational requirements. EfficientNet-B0 is highly efficient for mobile applications while EfficientNet-B7 achieves higher accuracy compared to ResNet-152 while using fewer FLOPS. Table 3.1 shows different EfficientNet variants, their parameters, and their performance metrics when trained on ImageNet.

U-Net is a fully convolutional neural network architecture specifically developed for overcoming problems encountered in biomedical image segmentation [81]. Traditional CNNs are designed for image classification whereas medical image segmentation requires

Table 3.1. Parameters and accuracy of EfficientNet models.

Model	Accuracy (ImageNet)	Parameters (Millions)	FLOPS (Billions)
EfficientNet-B0	76.3%	5.3	0.39
EfficientNet-B1	78.8%	7.8	0.70
EfficientNet-B2	79.8%	9.2	1.0
EfficientNet-B3	81.1%	12	1.8
EfficientNet-B4	82.6%	19	4.2
EfficientNet-B5	83.3%	30	9.9
EfficientNet-B6	84.0%	43	19.0
EfficientNet-B7	84.4%	66	37.0

precise pixel-wise classification where each pixel belongs to a class and there is limited labeled training data as annotated medical images are expensive and scarce. Thus, U-Net aims to provide efficient learning with small datasets and precise localization through skip connection. Furthermore, it performs faster segmentation by processing the whole image and better feature extraction by utilizing a symmetric encoder-decoder structure with a fully convolutional approach. It is named after its shape, as Figure 3.23 presents the U-shaped architecture for 32x32 pixels in the lowest resolution. In the image, blue boxes refer to multi-channel feature maps, and the number of channels is shown on top of the blue boxes while white boxes show the replicated feature maps. Furthermore, arrow colors represent different images, and the map for operations is shown at the bottom of the image.

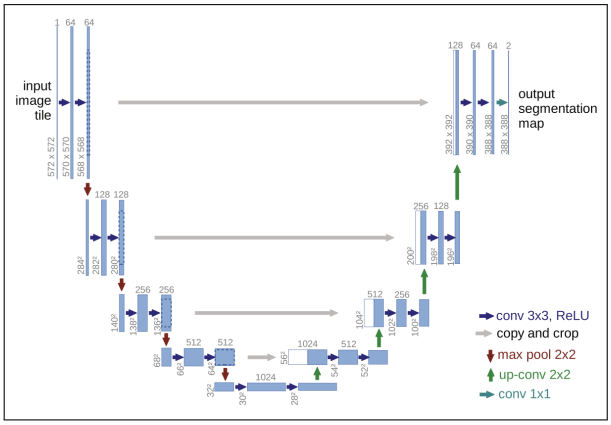


Figure 3.23. Architecture of U-Net [81].

U-Net architecture consists of two main components, namely contracting (encoder) and expanding (decoder) paths. While the encoder captures context and features, the decoder is assigned to restoring spatial information for segmentation, providing a symmetric structure.

The right side of shows the U-Net decoder structure while the left side of the figure refers to the encoder structure. Additionally, skip connections transfer fine-grained details from the encoder to the decoder, which prevents information loss due to downsampling and aids in precise localization. Its fully convolutional network allows input images of variable sizes as it does not comprise fully connected layers. As a loss function, U-Net utilizes a pixel-wise classification loss formula shown in Equation (3.23) where $p_{y(x)}$ is the predicted probability for pixel x and $w(x)$ is a weight map to handle class imbalance.

$$L = -\sum_{x} w(x) \log(p_{y(x)}) \tag{3.23}$$

Fast Region-based Convolutional Neural Network (Fast R-CNN) [82] is designed for performing object detection and classification computer vision tasks. Its novel feature involves making the entire detection pipeline trainable in a single network, instead of processing region proposals separately. This approach lets the region proposals be processed in a single forward pass, which makes it faster than regular R-CNN. Fast R-CNN consists of 4 main stages, namely feature extraction, region proposals, RoI pooling, and classification and bounding box regression. In feature extraction, an input image is passed through a CNN that is previously trained to extract feature maps. The pre-trained CNN is also referred to as the backbone. For the next process of proposing regions, Fast R-CNN directly pools features from proposals, instead of only conducting a selective search like traditional R-CNNs. Then, a RoI pooling layer extracts fixed-size feature vectors for each proposal. Lastly, the final fully connected layers classify objects while refining the bounding box coordinates. The complete pipeline for the Fast R-CNN structure is displayed in Figure 3.24 where the backbone of the architecture is VGG and fully connected layers output both classification and bounding box regression.

Faster R-CNN [83] on the other hand, is an improved version of Fast R-CNN by eliminating the region proposal bottleneck by introducing the Region Proposal Network (RPN), making the network significantly faster. The entire network structure is the same as Fast R-CNN such that Faster R-CNN is likewise a single unified network with feature extraction, RPN, RoI pooling, and final classification. RPN, however, generates object proposals directly from feature maps, thus, every proposal owns a bounding box and a ”objectness” score, which is

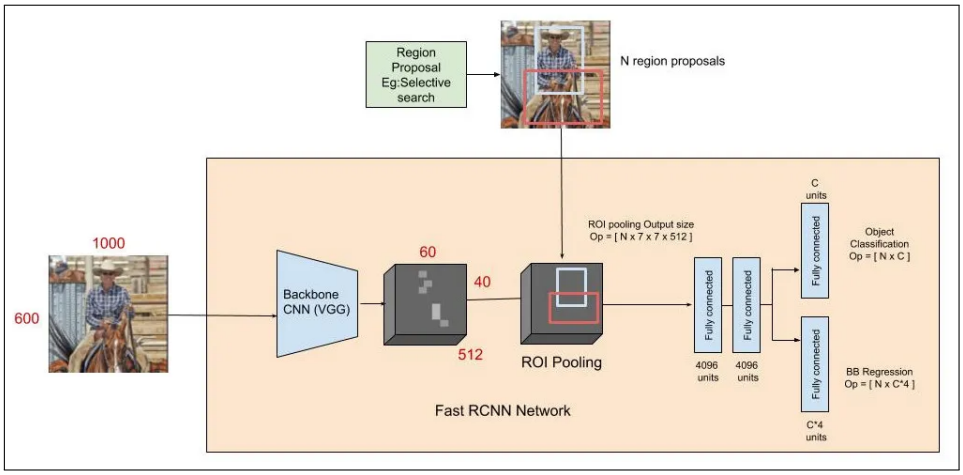


Figure 3.24. A sample Fast R-CNN pipeline [82].

the probability of containing an object.

Figure 3.25 illustrates the steps taken to process an image using the Faster R-CNN pipeline with an RPN. The process begins with the input image, shown at the bottom layer, being passed through convolutional layers for feature extraction, which produces a feature map. These feature maps capture essential spatial and semantic information. Then, the RPN slides over the feature map to generate region proposals by predicting the objectness score, which is used to check whether a known object is present in a region. It also predicts the bounding box coordinates to refine the proposal. The resulting proposals are visualized at the top of the figure as white rectangles. RoI pooling then converts each proposed region into a fixed-size feature map, ensuring a consistent feature representation regardless of region size. Finally, these features are passed to the fully connected network, the “classifier” in the figure, which assigns object classes.

The feature comparison of Fast R-CNN versus Faster R-CNN is presented in Table 3.2 in terms of region proposal method, processing speed, training time, and accuracy. End-to-end trainability in this table means that all parts of the DL model are trained simultaneously in a single optimization process, such that the input flows through the entire network, and all parameters are updated together using backpropagation and gradient descent.

Mask R-CNN [70] is an extension of Faster R-CNN that adds a segmentation branch to detect objects and simultaneously generate pixel-wise instance segmentation masks. While the output of a Faster R-CNN model is bounding boxes, the output of a Mask R-CNN model

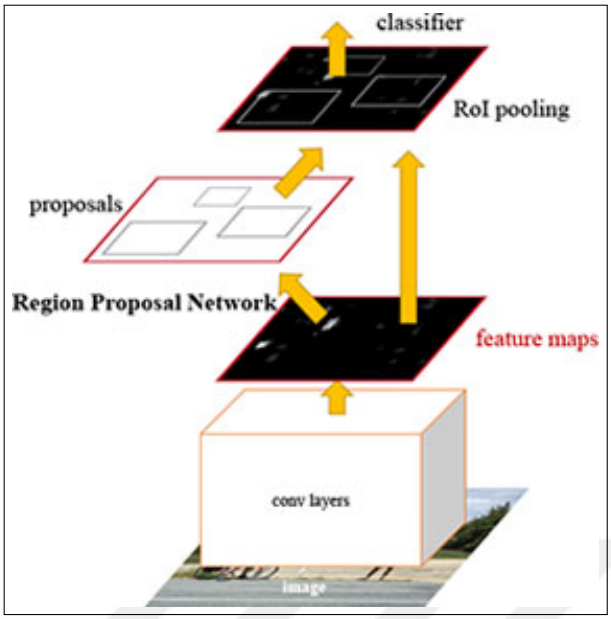


Figure 3.25. The region proposal procedure in Faster R-CNN [83].

Table 3.2. Comparison of R-CNN, Fast R-CNN, and Faster R-CNN

Feature	**R-CNN**	**Fast R-CNN**	**Faster R-CNN**
Region Proposal Method	Selective Search	Selective Search + RoI pooling	Region Proposal Network (RPN)
Speed	~14s per image	~0.2s per image	~0.02s per image
Feature Extraction	Extracts features for each region separately	Extracts features once for the entire image	Extracts features once for the entire image
Memory Efficiency	High	More efficient than R-CNN	Most memory efficient
Training Complexity	Multi-stage training	End-to-end training	End-to-end training
Bounding Box Regression	Separate model (trained after feature extraction)	Integrated into CNN	Integrated into CNN
Inference Time	Slow (Runs CNN multiple times per image)	Faster (Single CNN forward pass)	Fastest (RPN reduces proposal computation)
Accuracy	Good but slow	Better than R-CNN	Highest accuracy
End-to-End Trainability	**No** (Requires separate SVM for classification)	**Yes** (Single model)	**Yes** (Fully integrated)

is a segmentation mask. Figure 3.26a shows the output of a Fast R-CNN model detecting a fracture Figure 3.26b presents the output of Mask R-CNN model for the same radiograph.

The first component of Mask R-CNN is the feature extraction with a backbone network. This backbone network is commonly ResNet, which extracts the feature maps from the input image. Thereafter, the RPN generates bounding boxes for objects in the image. Instead of RoI pooling, Mask R-CNN employs RoI Align to maintain spatial details for achieving accurate mask predictions. Furthermore, it consists of three parallel output branches, namely classification, bounding box, and mask branch, as illustrated in Figure 3.27. The classification branch predicts the class of a given object while the bounding box regression branch refines the object locations. Mask branch, on the other hand, is only present in Mask R-CNN and it predicts the pixel-wise segmentation masks.

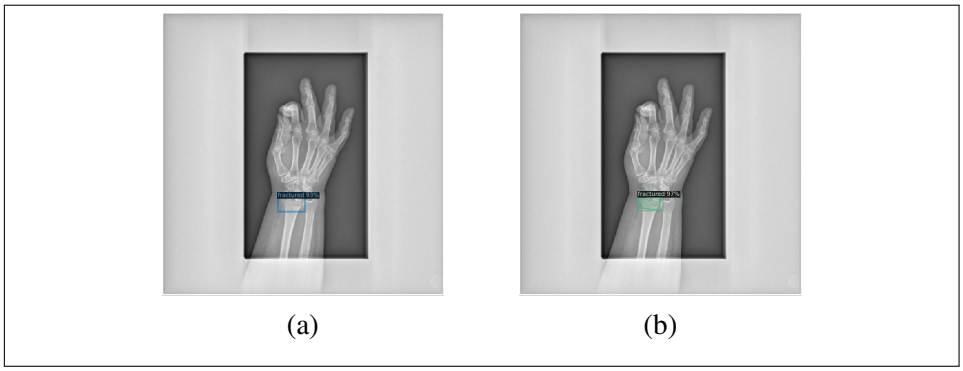


Figure 3.26. (a) Fast R-CNN prediction, (b) Mask R-CNN prediction.

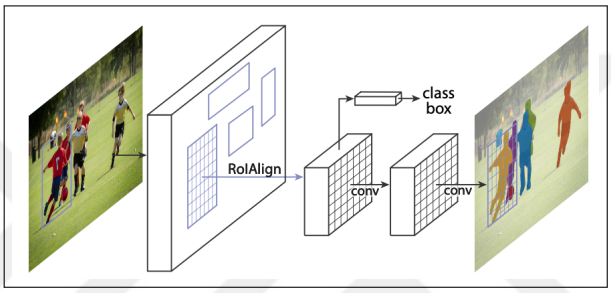


Figure 3.27. Mask R-CNN architecture for instance segmentation [70].

RoI Align is an improvement to RoI pooling utilized in Fast R-CNN and it solves the problem of spatial misalignment due to quantization errors while resizing the feature maps by utilizing bilinear interpolation. While RoI pooling rounds the coordinates, consequently causing spatial misalignment, RoI Align preserves the exact spatial locations. This approach provides an accuracy improvement in segmentation masks while improving the detection of smaller objects.

Since Mask R-CNN performs classification, bounding box detection, and mask prediction simultaneously, it employs a multi-task loss function to optimize each value concurrently. Compared to Faster R-CNN which only considers classification and bounding box regression losses, Mask R-CNN employs a mask loss component to ensure accurate instance segmentation. Equation (3.24) presents the multi-task loss function for Mask R-CNN where L_{cls} refers to classification loss computed using cross-entropy, L_{box} refers to the loss for the regression of the bounding box, calculated using $smooth_L$, and lastly, L_{mask} denotes the loss for segmentation mask calculated by binary cross entropy.

$$L = L_{cls} + L_{box} + L_{mask} \tag{3.24}$$

The cross entropy formula is given in Equation (3.25) to calculate the classification loss. It is employed to compare the predicted class probabilities against the ground truth, which ensures that the objects are classified properly while heavily penalizing the incorrect classifications. y_i is the ground truth label, which is valued at 1 if the classification is correct and 0 otherwise. Furthermore, $\hat{y}_i$ refers to the prediction probability for class i.

$$L_{cls} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3.25}$$

The bounding box regression loss, on the other hand, ensures that the predicted bounding boxes are as close as possible to the ground truth boxes. Mask R-CNN employs smooth L1 as it reduces the effects of outliers on the loss value, which is presented in Equation (3.26). While t_i denotes the predicted bounding box coordinates, $t^*{}_i$ refers to the ground truth bounding box coordinates. Equation (3.27) displays the adjustment of error penalization based on the magnitude. For small errors where x is smaller than 1, the function applies higher penalty, with an objective of providing smooth convergence. Contrarily, for larger errors, the function applies less penalty in order to prevent outlier impact by reducing gradient magnitude.

$$L_{\text{box}} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L1}(t_i - t_i^*) \tag{3.26}$$

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{3.27}$$

Lastly, the mask loss, which is unique to Mask R-CNN is employed to make sure that the model learns pixel-wise segmentation by applying binary cross-entropy loss at the pixel level and treating the mask prediction as a binary classification problem per pixel. The formula for this loss function is shown in Equation (3.28) where $m_{i,j}$ is the ground truth mask pixel value, which is 1 for pixels belonging to an object, and 0 for the background. $\hat{m}_{i,j}$, on the

other hand, denotes the predicted mask probability for pixel (i, j).

$$L_{\text{mask}} = -\sum_{i,j} \left[m_{i,j} \log(\hat{m}_{i,j}) + (1 - m_{i,j}) \log(1 - \hat{m}_{i,j}) \right] \tag{3.28}$$

While region-based object detection models achieve high accuracy, they depend on region proposals to detect objects. This has a tendency to make the training of these models expensive and not suitable for real-time applications as they first generate region proposals and then classify the objects. Thus, to address these challenges, You Only Look Once (YOLO) [84] presents a single-stage object detection framework that eliminates the requirement of region proposals, directly predicting the object locations and class probabilities in a single forward pass, looking at the image only once, hence the name. While Faster R-CNN processes multiple region proposals per image, YOLO treats the object detection procedure as if it is a regression problem.

Specifically, instead of scanning an image with a sliding window or generating region proposals, YOLO splits the image into a grid and performs prediction per cell in the grid. Presented in Figure 3.28, the input image is divided into an SxS grid and if an object's center is inside the grid cell, that particular grid cell is responsible for detecting the object. Every grid cell is assigned to predict bounding boxes, the confidence score for each bounding box, denoting how likely the box contains an object and its Intersection over Union (IoU) with the ground truth, and class probabilities.

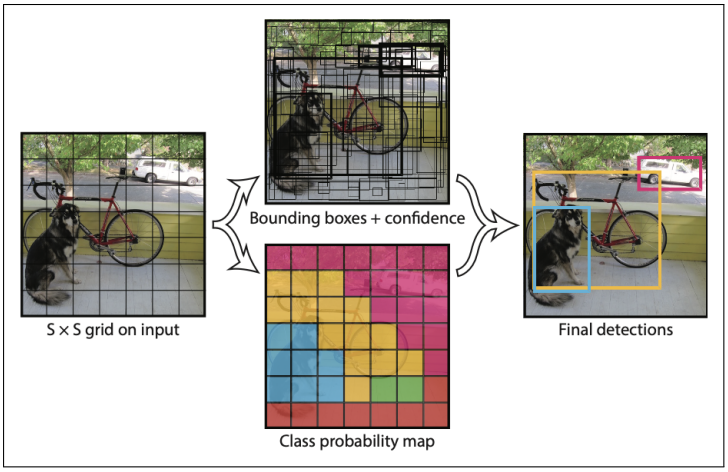


Figure 3.28. YOLO architecture [84].

Each prediction made by the network is stored in a tensor, encoded as presented in Equation (3.29) where B is the bounding box present in the grid and C is the class probabilities. Each bounding box, B in Equation (3.29), has coordinates as x, y, width, height, and confidence. While the x and y are center coordinates relative to thee grid cell, width and height are relative to the image.

$$\text{Final output tensor} = S \times S \times (B \times 5 + C) \tag{3.29}$$

Figure 3.29 illustrates the overall structure of the YOLO network, presenting how an input image is processed through the architecture to output final object detections. In this figure, the YOLO network resizes an input image with a resolution of 448x448 with 3 channels of color, red, green, and blue prior to feeding it to the network. The first 24 convolutional layers work to extract spatial features from the image, such as edges, textures, and object shapes. For this procedure, before applying 3x3 convolution layers, 1x1 convolutions are employed to reduce the depth, which captures the hierarchical features that convert simple textures to complex object structures. This structure utilizes Leaky ReLU as the activation function while using max pooling layers to perform the reduction of spatial dimensions.

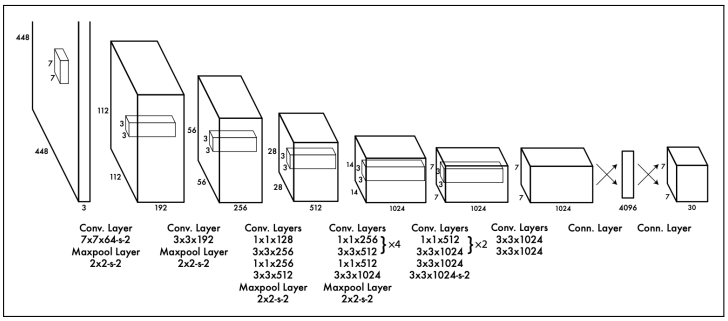


Figure 3.29. YOLO network structure [84].

The last two layers are the fully connected layers in the YOLO network. Once the feature extraction process is complete, the feature map is flattened into these connected layers that predict bounding boxes and class probabilities. The resulting predictions include bounding box coordinates and the probability for the object category. The last two layers are the fully connected layers in the YOLO network. Once the feature extraction process is complete, the feature map is flattened into these connected layers that predict bounding boxes and

class probabilities. The resulting predictions include bounding box coordinates and the probability for the object category. Unlike region-based models, YOLO performs object detection using a single forward pass over the entire image. It processes the image through a convolutional network to extract features, which are then passed to fully connected layers to predict bounding boxes and class probabilities in one streamlined step. Furthermore, YOLO is end-to-end trainable, meaning that the network learns object detection directly from images without separate region proposal steps.

YOLO9000, also known as YOLOv2, improved upon the initial YOLO by providing higher accuracy [85]. It is capable of detecting over 9000 object categories by combining detection and classification data, making it a multi-scale and multi-label object detector. The key innovation of YOLOv2 is that it is jointly trained on both the Common Objects in Context [86] dataset for object detection and the ImageNet [87] dataset for classification, allowing it to detect objects despite not having labeled bounding boxes. Compared to the initial YOLO, this version is capable of detecting a wider variety of objects while achieving high accuracy.

This network introduces batch normalization, which stabilizes the training while improving the accuracy. Furthermore, YOLOv2 predicts the bounding boxes using anchor boxes instead of coordinate regression. Anchor boxes refer to the predefined bounding boxes to help the model handle scale and aspect ratio variations, which means that the YOLOv2 model adjusts the anchor box to fit the detected object instead of predicting object coordinates directly, which makes the detection process more stable. Additionally, YOLOv2 employs dimension clustering to determine better anchor box priors using k-means clustering. **??** displays the clustering box dimensions on the Pascal Visual Object Classes (VOC) [88] and COCO datasets. Figure 3.30a shows the mean IoU received from experimenting with various choices for the k value in k-means clustering, which reveals that k valued at 5 returns the best metrics. Figure 3.30b the image, on the other hand, shows the relative centroids for both of the datasets. Moreover, YOLOv2 introduces using passthrough layers to maintain high-resolution features for better small object detection. Lastly, YOLOv2 is trained at random input resolutions, making the overall model robust to different image sizes, providing multi-scale training.

YOLOv3 is an improved version of YOLOv2, enhancing accuracy with better feature extraction while maintaining speed [89]. It detects objects at three different scales, improving

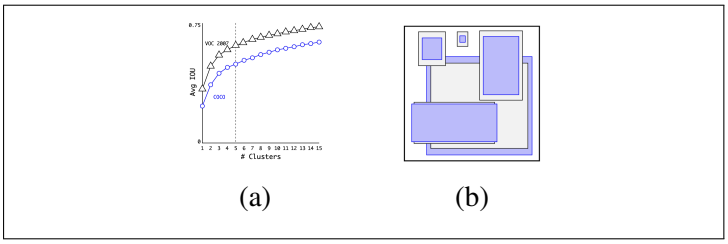


Figure 3.30. (a) YOLOv2 mean IoU, (b) box dimension clustering [85]

performance on small objects. YOLOv3 introduces a Feature Pyramid Network (FPN), which combines features from different layers to handle varying object sizes. YOLOv4 builds on this with a new backbone and the Path Aggregation Network (PAN), reducing computational cost and improving gradient flow. It achieves 43.5% mAP on the COCO dataset at 62 FPS, outperforming YOLOv3's 33% mAP at 20 FPS [90].

YOLOv5, developed by Ultralytics [91], shifts to PyTorch for easier development and integration. It introduces various model sizes, from YOLOv5n (smallest, fastest) to YOLOv5x (largest, most accurate), allowing users to balance speed and performance based on task requirements. Key features include advanced data augmentation, auto-learning bounding box anchors, and performance optimization. Additional key features of YOLOv5 include enhanced data augmentation, auto-learning bounding box anchor, and performance optimization. It introduces mosaic augmentation during training, which works by combining 4 images into one during training, which improves the model's ability to detect objects in varied contexts while enhancing robustness to different object scales and orientations.

YOLOv8, likewise developed by Ultralytics [92], is a significant leap forward in object detection and computer vision, offering a blend of speed, accuracy, and versatility that caters to a wide range of applications. It introduces several architectural enhancements and improved performance metrics, making it a state-of-the-art (SOTA) solution for various computer vision tasks. Figure 3.31 represents the complete YOLOv8 architecture, comprised of its backbone, neck, and head.

The backbone in YOLOv8 uses Cross Stage Partial Networks (CSP) to extract multi-scale feature maps from input images. Features are processed through five hierarchical stages (P1 to P5), where lower levels capture fine details and higher levels extract semantic information. These stages enable efficient multi-scale feature learning while reducing

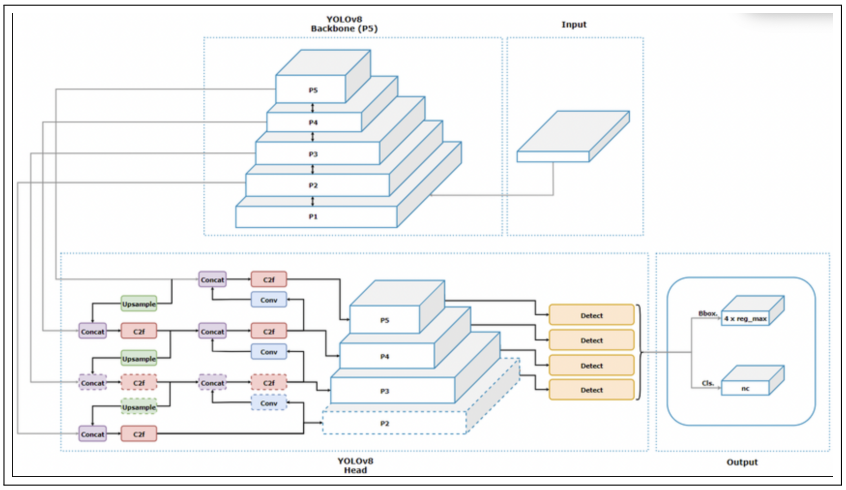


Figure 3.31. YOLOv8 architecture [93].

redundant computation. The neck incorporates the Path Aggregation Network (PANet) to refine and merge features across layers using upsampling, concatenation, and convolution operations. The head performs object detection and classification, producing bounding boxes (BBox) and class probabilities (Cls) from the processed features. Unlike earlier YOLO versions that rely on anchor boxes, YOLOv8 adopts an anchor-free approach by directly predicting object centers, which simplifies detection and improves accuracy. Its architecture supports tasks such as real-time detection, segmentation, and classification.

YOLOv11 is the latest generation of the YOLO object detection series, developed by Ultralytics [94]. It builds upon the existing YOLOv8 architecture, introducing improved efficiency, accuracy, and computational performance. YOLOv11 replaces the C2f block from the previous iteration, YOLOv8 with the C3k2 module, which is a more efficient CSP bottleneck. Rather than the one large convolution utilized in C2f, C3k2 employs two convolutional layers, which improves efficiency while maintaining accuracy, resulting in an enhanced feature extraction and gradient flow while reducing computation requirements. Figure 3.32 illustrates the overall architecture of YOLOv11, featuring the new C3k2 blocks.

As revealed in the Figure 3.32 as well, YOLOv11 utilizes C2PSA blocks, which combines CPS with spatial attention. This utilization improves spatial attention in feature maps, especially enhancing small object detection as well as overlapping object separation in crowded images. The head structure remains similar to that of YOLOv8, still utilizing the anchor-free approach.

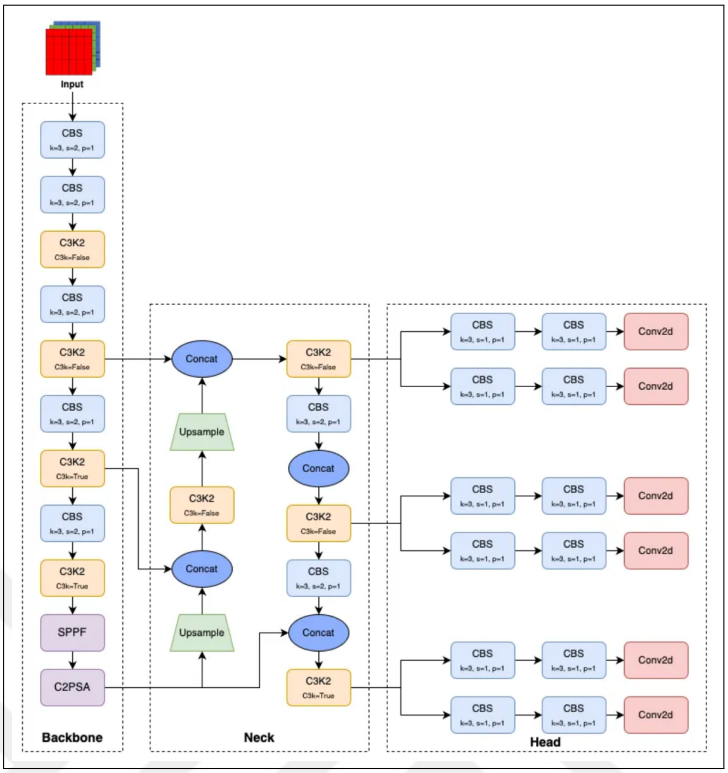


Figure 3.32. YOLOv11 architecture [93].

3.4. MODEL EVALUATION METRICS

Evaluation metrics are crucial to assess the models' effectiveness, efficiency, and generalization capabilities. While different models introduce architectural enhancements to improve detection accuracy, their performance must be quantitatively evaluated using standardized metrics. This section presents key evaluation metrics used in object detection, segmentation, and classification tasks, providing insight into model performance, accuracy, and computational efficiency. Understanding these metrics is essential for comparing models and determining their suitability for real-world applications.

Robust evaluation metrics are required to quantify model accuracy and compare different architectures, assess localization performance for bounding boxes, detect biases or errors in model predictions, ensure generalizability to real-world data, optimize hyperparameters, and guide improvement during model development. Good evaluation metrics should answer questions such as how well the model detects and classifies objects, is the model capable of detecting all relevant objects while avoiding false negatives, and how precise are the predicted

bounding boxes or masks [95]. Evaluating different models presents unique challenges beyond standard classification tasks. Models are often required to balance catching all objects and avoiding false positives, achieving a trade-off between them. For different tasks, evaluation objectives vary. For instance, in medical imaging, catching any object is crucial to prevent overlooking critical abnormalities. When it comes to autonomous driving, on the other hand, it should be avoided to incorrectly identify objects on the road, seeking to prevent false detections. Furthermore, the trade-off between the computational cost versus the accuracy should be maintained, since faster models may sacrifice some accuracy for real-time performance while more accurate models are more computationally expensive, requiring a balance between speed and precision.

A confusion matrix is a fundamental tool for evaluating classification models. It provides a detailed breakdown of a model's predictions compared to the actual ground truth, helping measure accuracy, precision, recall, and F1-score [95]. A confusion matrix is commonly a 2x2 table that categorizes predictions into four groups, namely true positive (TP), false positive (FP), true negative (TN), and false negative (FN), as illustrated in Figure 3.33.

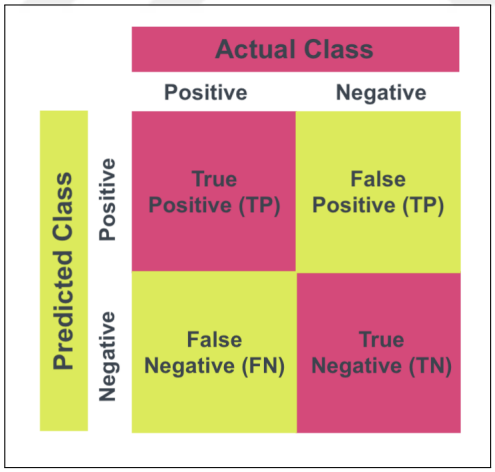


Figure 3.33. Confusion matrix.

TP occurs when the model correctly predicts an object when one is present, FP, on the other hand, happens if the model incorrectly detects an object when there is none [95]. FN transpires when the model fails to detect an object that is actually present. TN ensues if the model correctly predicts no object when there is none. Accuracy measures the overall correctness of a classification model by calculating the percentage of correct predictions overall predictions made. It is one of the most basic evaluation metrics used in machine

learning, particularly for classification tasks. The formula for calculating accuracy is presented in Equation (3.30). High accuracy means most predictions are correct, however, it is not always reliable for imbalanced datasets where one class is significantly outnumbered by another class. Furthermore, if an object detection model misses all small objects, but correctly detects large objects, the accuracy might still appear high. For instance, in medical imaging, if 98% of all images do not contain a fracture, a model that always predicts ”No Fracture” would have 98% accuracy but be completely useless.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.30}$$

Precision, also known as positive predictive value, measures how many of the predicted objects are actually correct [96]. The formula for precision is presented in Equation (3.31) which is the ratio of TP to all positive predictions. While high precision means fewer FPs, maintaining a trade-off between precision and recall is important. A model with high precision may have low recall, meaning it misses objects. For instance, in autonomous driving, precision should be prioritized to avoid false alarms.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.31}$$

Recall, on the other hand, calculates how many actual objects the model correctly detects [96]. It is also known as sensitivity. While high recall ensures that the model does not miss important objects, it may result in more false positives, resulting in low precision. The formula for recall is shown in Equation (3.32) and it should be prioritized in the medical domain where missing a real positive is critical.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.32}$$

F1-Score is of utmost importance to evaluate the model’s applicability as it provides a balance between precision and recall. Precision and recall might often conflict, requiring a trade-off, the F1-score balances them by taking the harmonic mean between the two metrics as presented in Equation (3.33). It is particularly useful as it provides a single metric that

balances false positives and false negatives. It can prove to be especially valid in imbalanced datasets where accuracy is misleading.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.33}$$

Receiver Operating Characteristic (ROC) and Area Under Curve (AUC-ROC) are employed in binary classification tasks where the models predict probabilities instead of fixed labels. ROC curves visualize the trade-off between the TP rate (recall) and FP rate, which is computed in Equation (3.34).

$$\text{False Positive Rate} = \frac{FP}{FP + TN} \tag{3.34}$$

Figure 3.34 shows the sample ROC curve for a fictional training. TPR, which is sensitivity is shown on the Y-axis, which measures how many actual positives are correctly identified. FPR shown on the X-axis, on the other hand, measures how many negative cases are incorrectly classified as positive. In this ROC curve, the curve is too close to the diagonal, grey dashed line, which suggests that the classifier performs poorly and is close to random guessing. A perfect classifier would have a curve that is close to the top left corner such that TPR is closer to 1 and FPR is closer to 0. AUC ranges from 0 to 1 where if it is equal to 1, this refers to perfect classification capabilities. If it is equal to 0.5, the model performs random guessing, meaning it has no predictive power. If it is smaller than 0.5, this means that the model is worse than random such that it is misclassifying more often than chance. In this graph, AUC is equal to 0.4, which suggests that the model is actually worse than random in its predictions. AUC-ROC is particularly useful for detecting the imbalance between positive and negative samples. Furthermore, it helps compare different models before selecting a classification threshold.

The Precision-Recall (PR) curve is another effective evaluation tool for models dealing with imbalanced datasets, where positive cases are rare. It helps visualize the trade-off between precision which is the positive predictive value and recall, which is the model's sensitivity to TP across different classification thresholds. Figure 3.35 shows a sample PR curve after a

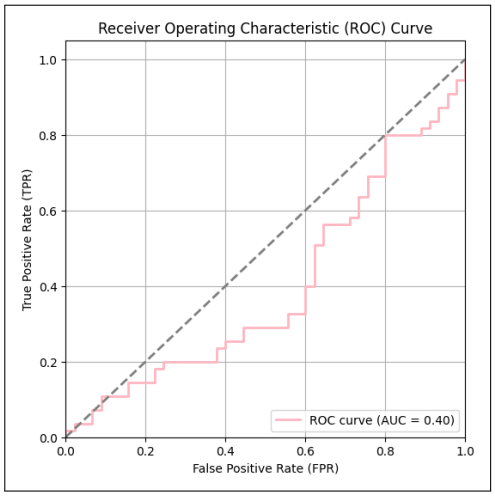


Figure 3.34. A sample AUC-ROC curve for a fictional training.

fictional training. While the X-axis shows recall, the Y-axis denotes precision. It operates by assigning a probability score to each sample. By changing the decision threshold, the model can classify a sample as positive or negative at different confidence levels. The PR Curve shows how Precision and Recall change as the threshold varies. A good model needs to have high precision and high recall, which means that it detects most positive cases without many false positives. However, a random model has a PR curve near the baseline precision.

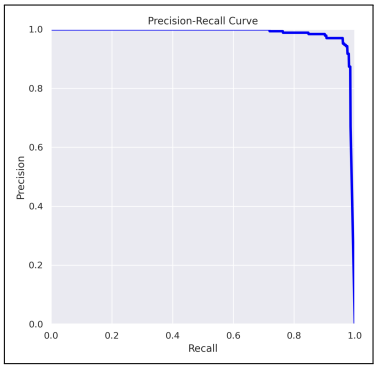


Figure 3.35. A sample PR curve for a fictional training.

Intersection over Union (IoU) is a fundamental metric in object detection that quantifies how well a predicted bounding box overlaps with the ground truth bounding box. It is calculated with the formula given in Equation (3.35). While higher IoU means better localization accuracy, low IoU indicates poor overlap between the prediction and the ground truth.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{3.35}$$

Figure 3.36 illustrates the IoU such that the red box is the predicted bounding box by the model, and the yellow box is the ground-truth, actual bounding box. The overlapped area is denoted in orange and this area is calculated by the formula in Equation (3.35). IoU is used particularly in Non-Maximum Suppression (NMS) that filters out the overlapping detections, mean average precision (mAP) which evaluates the overall model accuracy, and in segmentation tasks where IoU is extended to the Jaccard Index.

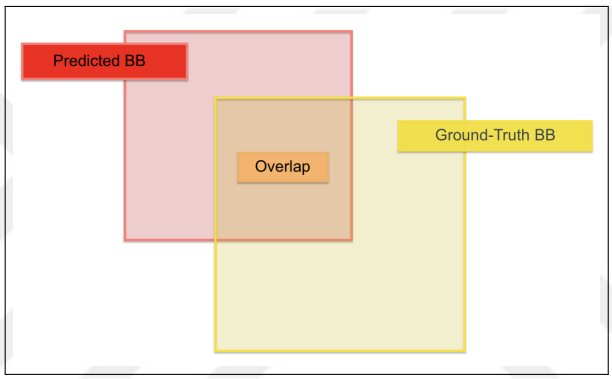


Figure 3.36. Illustration of IoU.

Object detection models do not just classify objects; they also accurately localize them. Mean Average Precision (mAP) is the gold standard for evaluating detection models such as YOLO, Faster R-CNN, and Mask R-CNN. It summarizes both classification accuracy and localization accuracy using IoU thresholds. Average precision (AP) combines precision and recall across multiple confidence thresholds, measuring how well a model can balance precision and recall over different levels of IoU. AP is calculated as the area under the PR curve, which is presented in Equation (3.36). Higher AP yields better object detection accuracy.

$$AP = \int_0^1 P(R)dR \tag{3.36}$$

AP@50 is the Pascal VOC benchmark standard where average precision is computed when the IoU threshold is set to 0.5 [95]. For this metric, a detection is considered to be correct if the overlap between the ground truth and the bounding box is larger than 0.5. AP@50 is

more lenient compared to AP as it accepts bounding boxes that overlap at least 50% with the ground truth. AP@50:95, on the other hand, is the main evaluation metric for COCO benchmarks. Instead of using the only threshold of 0.5 for IoU, COCO benchmarks require models to compute AP at multiple IoU thresholds, ranging from 0.5 to 0.95 in steps of 0.05, as shown in Equation (3.37). Thus, this metric evaluates both detection and precise localization while rewarding the models that localize objects with high precision.

$$AP@50:95 = \frac{1}{10} \sum IoU_{0.50}^{0.95} AP(IoU) \tag{3.37}$$

Lastly, mean average precision (mAP) is the mean of all AP values across classes and IoU thresholds, calculated as presented in Equation (3.38) where N is the number of classes. mAP can effectively summarize the overall performance of a model across multiple object categories. Interpreting mAP requires understanding the trade-offs between AP@50 and AP@50:95. A model with high AP@50 but low AP@50:95 suggests that it is good at detecting objects but struggles with precise localization. A higher AP@50:95 score indicates a strong model that consistently places bounding boxes with high accuracy, which is crucial in applications like medical imaging and autonomous driving where precision is critical.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3.38}$$

In addition to detection metrics, there are also evaluation methods specific to segmentation tasks. The Dice Coefficient, also known as the Sørensen-Dice Index, is a commonly used evaluation metric for image segmentation tasks. It quantifies the similarity between the predicted segmentation mask and the ground truth mask by measuring their overlap. The Dice Coefficient is calculated using the formula in Equation (3.39) where A is the set of pixels in the predicted segmentation mask, B is the set of pixels in the ground truth mask, ∩B represents the intersection of the two masks which are the overlapping pixels, and $|A| + |B|$ represents the total number of pixels in both masks. While a higher Dice score indicates better segmentation accuracy, the Dice coefficient ranges from 0 to 1 where if D is valued at 1.0, there is perfect overlap between predicted and ground truth masks and if D is valued at 0.0, there is no overlap between two.

$$D = \frac{2 \times |A \cap B|}{|A| + |B|} \tag{3.39}$$

The Jaccard Index, also known as the Intersection over Union (IoU) for segmentation, is a key metric for evaluating the accuracy of image segmentation models. Similar to the Dice Coefficient, it measures the overlap between a predicted segmentation mask and the ground truth mask, but it does so in a slightly different way as it measures the fraction of pixels correctly classified out of all pixels that were labeled as either positive in the prediction or ground truth. Similar to the Dice score, the value ranges from 0 to 1 while higher Jaccard values indicate better segmentation accuracy. Jaccard index directly quantifies segmentation accuracy by penalizing false positives and false negatives and unlike accuracy, it is robust to class imbalances, since background pixels do not affect the score. The computation for the Jaccard index is shown in Equation (3.40), measuring the ratio of intersection to the total unique pixels in both masks. The difference between Dice and Jaccard is presented in Equation (3.41) and since Dice gives twice the weight to intersection, it generally yields higher values than Jaccard.

$$J = \frac{|A \cap B|}{|A \cup B|} \tag{3.40}$$

$$D = \frac{2J}{1 + J} \tag{3.41}$$

While accuracy-based metrics are crucial for evaluating the quality of segmentation models, they do not account for a model's computational efficiency. In real-world applications, especially those requiring real-time inference, it is not enough for a model to be accurate; it must also be fast and efficient.

Deep learning models, particularly CNN-based architectures like YOLO, Faster R-CNN, Mask R-CNN, and U-Net, can vary significantly in their computational cost. Some models prioritize speed (e.g., YOLO), while others focus on detection accuracy (e.g., Faster R-CNN). To assess how efficiently a model runs in deployment, computational performance metrics are utilized, including Frames Per Second (FPS), Inference Time (Latency per Image), and Floating Point Operations per Second (GFLOPs).

FPS measures how many images a model can process per second. It is a critical metric for

real-time applications, where high-speed inference is essential. It is calculated as presented in Equation (3.42). For instance, if a model takes 0.02 seconds per image, its FPS is 50 frames per second. A higher FPS means faster inference, making the model suitable for real-time use cases. YOLO models have high FPS while Mask R-CNN has lower FPS because of its complex segmentation tasks.

$$FPS = \frac{1}{\text{Inference Time per Image}} \tag{3.42}$$

Inference time, on the other hand, measures how long it takes for a model to process a single image and it is the inverse of FPS. For applications where real-time responsiveness is critical, low latency is preferred. Lastly, GFLOPs measure the computational complexity of a deep learning model. A model with higher GFLOPs requires more processing power, often requiring Graphical Processing Units (GPUs) or Tensor Processing Units (TPUs) but yielding better accuracy. Low GFLOP models are better for mobile devices and edge computing.

4. ANALYSIS AND DESIGN

The following section outlines the fundamental requirements and architectural considerations to develop a robust fracture detection and segmentation system. Given the complexity of radiographic images and the necessity for precise localization of fractures, the system shall be designed to optimize accuracy while ensuring reliability in real-world clinical scenarios.

Furthermore, this section discusses the functional and non-functional requirements of the proposed system, by listing the core functionalities required for data preprocessing, model training, evaluation, and real-world testing. Additionally, provides details regarding the envisioned application. By these requirements, this section also addresses system usability, performance constraints, and security measures for handling medical data.

Following the requirements analysis, the architectural design of the system is presented using various diagrams to illustrate its workflow, structure, and interactions. These include the use case diagram, which visualizes how different users interact with the system; the class diagram, which represents the system's data structure; the sequence diagram, which illustrates the step-by-step flow of operations; and the component diagram, which showcases the high-level system architecture.

4.1. SYSTEM REQUIREMENTS

Functional requirements define the targeted functionalities that the system aims to achieve, along with the specific sub-tasks it must support. The functional requirements of this study are categorized into three key phases: model training, comparison, and application development.

For data processing, model training, and comparison, the functional requirements are as follows.

- The approach must preprocess input radiographs by normalizing pixel intensities, resizing images, and applying augmentations to improve model robustness.
- The experimental pipeline must support the training of multiple detection and segmentation models, including Faster R-CNN, YOLOv8, YOLOv11, Mask R-CNN,

and EfficientNet.

- The approach must support validation during training, utilizing metrics such as mAP, precision, recall, F1-score, and IoU.
- The framework must store training logs and model checkpoints for further analysis and reproducibility.
- The approach must evaluate trained models on an independent real-world dataset annotated by orthopedic surgeons.
- The approach must analyze false positives and false negatives, categorizing errors into various types.
- The evaluation pipeline must compare models based on inference time, accuracy, and robustness, highlighting strengths and weaknesses in real-world medical imaging.
- The approach must generate visual representations of model predictions, including bounding boxes for detection models and segmentation masks for segmentation models.
- The evaluation framework must allow orthopedic surgeons to review model outputs and provide qualitative feedback.
- The final comparative study must compare the best-performing model against medical interns, using orthopedic surgeons' annotations as the ground truth standard.
- The framework must record and analyze human versus AI performance, assessing accuracy, processing time, and error types.

For the application, on the other hand, the functional requirement are as follows.

- The system must allow users to upload radiographs for fracture detection or segmentation.
- The system must run inference on uploaded images using the best-performing model from the comparative analysis.
- The system must provide side-by-side visualization of the original image and predicted

output with an adjustable colormap.

- The system must display processing time and a confidence score for each prediction.
- The system must allow users to save the segmented output as an image file.
- The system must include a feedback mechanism where doctors can submit feedback before or after running the prediction.
- The system must store feedback as a .log file, including metadata such as timestamp, image name, and feedback content.

Non-functional requirements should define quality attributes for the envisioned approach. The non-functional requirements of this study are categorized into two key areas once more: model training, comparison, and application development.

In this section, approach refers to the overall experimental and application framework that incorporates model training, evaluation, feedback mechanisms, and performance analysis. The non-functional requirements for the model training and comparison are as follows.

- The approach should enable models to process radiographs in under 5 seconds per image to ensure real-time feasibility.
- The approach should support large-scale datasets without performance degradation.
- The approach should be optimized for GPU acceleration, reducing inference and training time.
- The approach should be adaptable to integrate new datasets and model architectures without major modifications.
- The approach must ensure consistency across evaluations, maintaining standardized preprocessing, hyperparameters, and evaluation metrics.
- The evaluation methodology must ensure that results are reproducible and comparable across different runs.
- The approach must ensure that preprocessing, training, and evaluation steps are well-documented for future studies.

- The system must uphold data confidentiality by enforcing secure access protocols for sensitive inputs and predictions

The non-functional requirements for the application are listed below.

- The system must provide an intuitive interface that is easy to use for non-technical users, including medical professionals.
- The system must prioritize clear and visually interpretable outputs, allowing users to easily assess model predictions.
- The system must be designed with a simple workflow that enables users to upload images, run predictions, and view results seamlessly.
- The system must allow doctors to submit feedback in under 15 seconds and through no more than 5 user interactions, before or after running the prediction.
- The system must perform local inference in under 2 seconds on GPU per image, ensuring usability in real-world scenarios.
- The system must handle multiple image formats without errors.
- The application must store feedback reliably in log files, ensuring that no data is lost during submission.
- The application must store feedback in encrypted form using the Advanced Encryption System (AES) method before appending it to a log file. Each entry must be uniquely timestamped and saved without overwriting previous submissions.
- The system must handle radiographic data ethically and securely by ensuring encryption during storage, controlled access, and avoidance of personally identifiable information.

4.2. SYSTEM WORKFLOW AND RESEARCH PIPELINE

The diagram presented in Figure 4.1 illustrates a structured overview of the research workflow for DRF detection. The phases are grouped under three main stages, namely data preparation,

model training, and evaluation and deployment. Overall, the structure reflects the methodical approach that emphasizes the scientific rigor of model development and the eventual goal of real-world applicability.

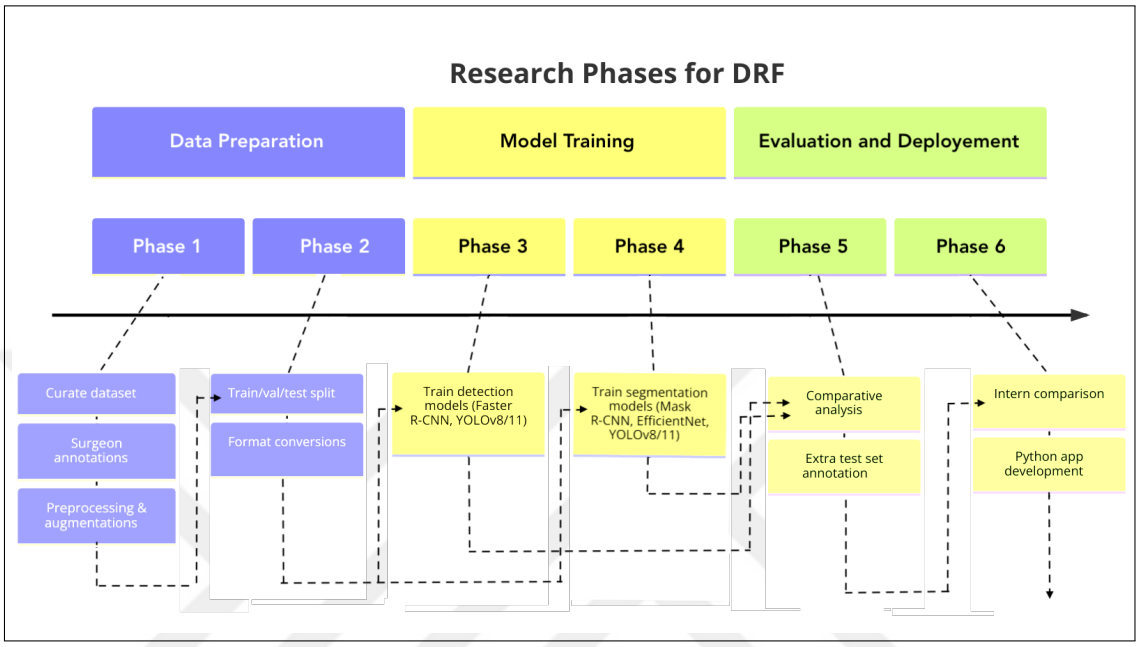


Figure 4.1. The general workflow of the proposed study.

The data preparation stage encompasses 2 phases. While phase one focuses on curating a high-quality dataset, which includes collecting radiographic images, acquiring annotations from orthopedic surgeons, and applying necessary preprocessing and augmentation techniques, phase two continues this preparation with dataset partitioning into training, validation, and test sets, alongside format conversions that ensure compatibility with different model architectures, which ensures a standardized and fair basis for all subsequent evaluations.

Following data preparation, the workflow progresses into the model training stage, consisting of Phase 3 and Phase 4. Phase 3 involves the training of detection models, specifically Faster R-CNN and the YOLO family (v8 and v11), which are designed to locate fractures using bounding boxes. Phase 4 shifts focus to segmentation models, including Mask R-CNN, EfficientNet, and segmentation-capable variants of YOLOv8 and YOLOv11. The separation of detection and segmentation training phases allows for a comparative assessment of these fundamentally different approaches in identifying and localizing fractures.

The final stage, evaluation and deployment, comprises Phase 5 and Phase 6. In Phase 5, a comparative analysis is conducted to evaluate the performance of all trained models. This involves not only standard metrics but also real-world validation using an additional, independently annotated test set. This extra layer of evaluation strengthens the external validity of the study. Phase 6 marks the application-focused culmination of the project. Here, the best-performing model, which is selected based on both metric-based evaluation and doctor preference, is tested against the findings of medical interns. This phase also includes the development of a Python-based application to deploy the model in a usable form, complete with inference visualization and a feedback mechanism for medical professionals to use.

Furthermore, the flowchart presented in Figure 4.2 displays the steps that need to be taken to design the DRF prediction approach, outlining the full pipeline from data preparation to model deployment. Accordingly, the work begins with creating a radiograph dataset, obtaining expert annotations, and performing preprocessing and augmentation to prepare the data. After splitting the dataset and ensuring consistent formatting, the process branches into training both detection and segmentation models. These models are then compared, not only on validation data but also on an additional real-life test set with expert annotations. To assess real-world relevance, the best-performing model is compared against medical interns using surgeon-labeled ground truths.

Figure 4.3 presents the Gantt chart of the entire research steps for the developement of a deep-learning based sytem for DRF detection and segmentation. The visualization supports a systems-level understanding of the workflow, highlighting both the chronological and logical relationships between each element. Once the models are trained, they generate predictions which are forwarded to the evaluation component. Evaluation plays a central role in the comparative analysis of model performance. Ground truth annotations for an additional test set that are sourced independently from a hospital are again provided by orthopedic surgeons, ensuring that model assessment is aligned with expert judgment.

The evaluation component synthesizes the model predictions, surgeon annotations, and intern outputs to provide feedback on model performance. This feedback loop can guide further model refinement, if necessary. Once the most suitable model is identified, it is exported and integrated into a standalone Python application designed to perform DRF detection

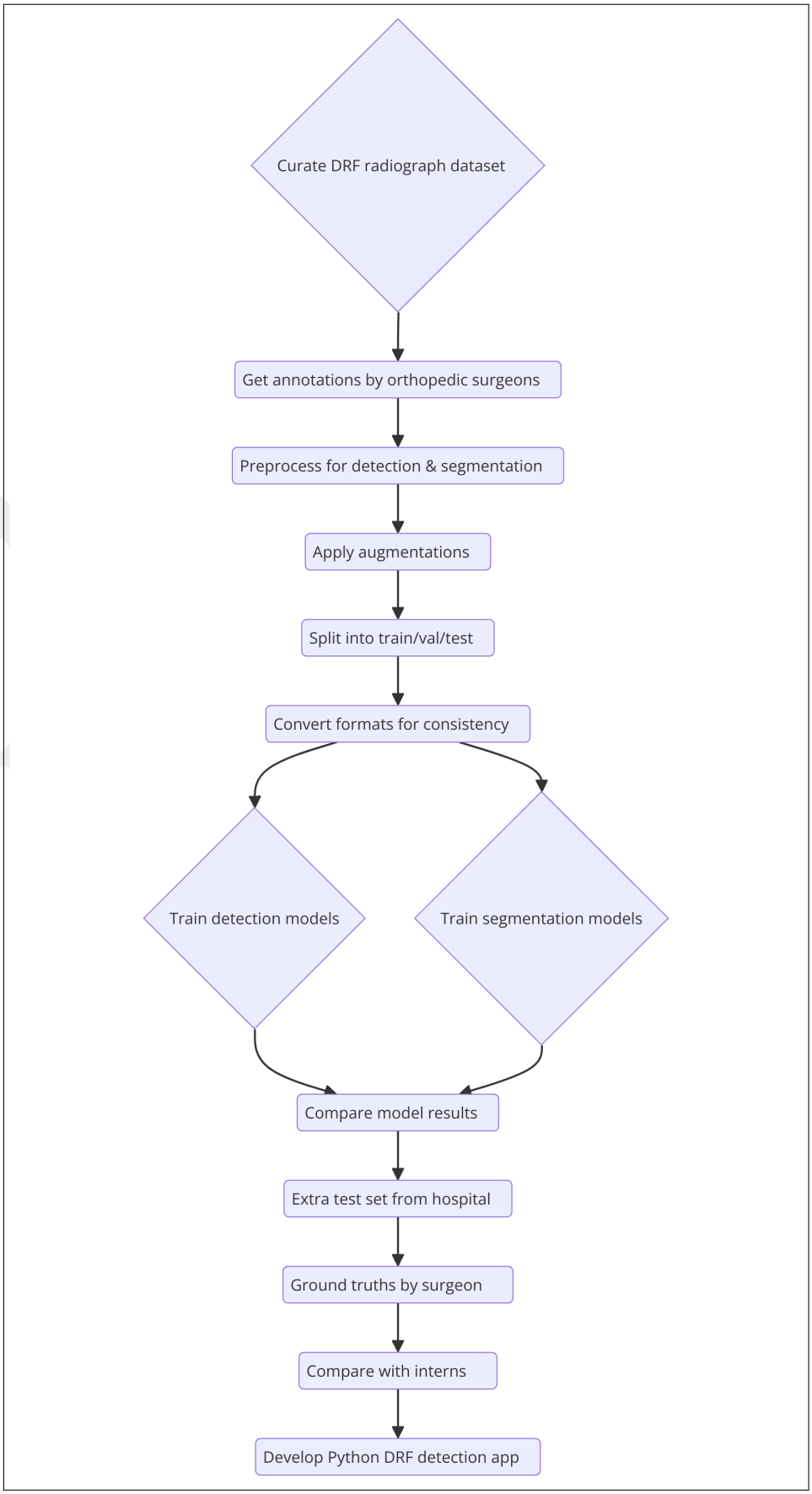


Figure 4.2. The detailed methodology flow of the proposed study.

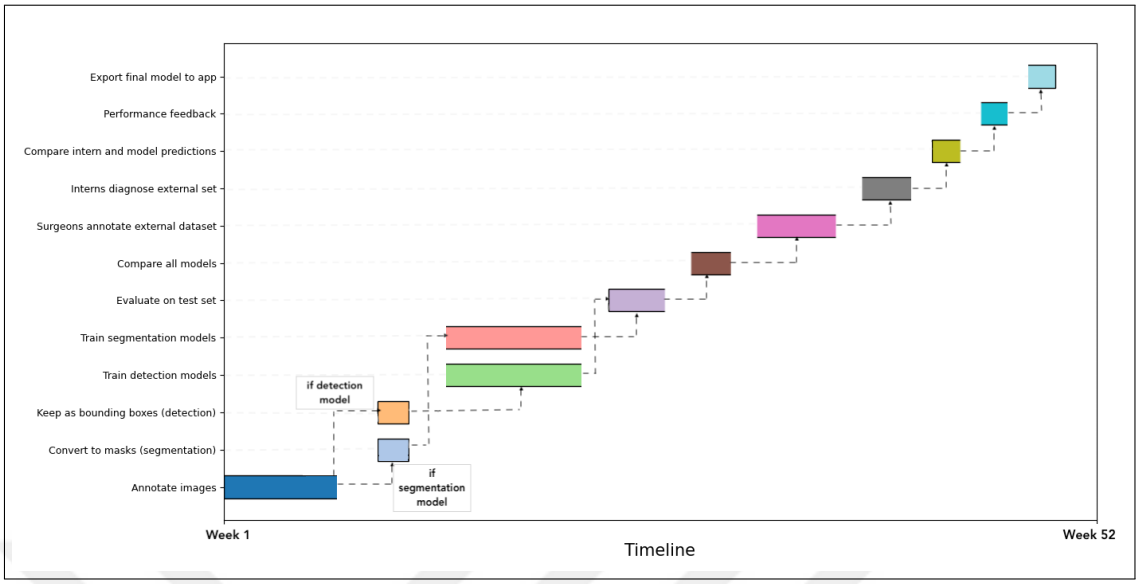


Figure 4.3. The Gantt chart for the study.

across various image inputs. The application serves as the end-user interface, delivering model-driven results to clinicians and other users.

Notably, the sequence diagram in Figure 4.3 encapsulates the end-to-end nature of the study, transitioning from data collection and annotation to model training, validation, and deployment. By incorporating a Python-based application at the conclusion of the pipeline, the study translates its findings into a practical tool, capable of assisting in real-world diagnostic workflows. This comprehensive interaction model reflects a research design grounded in reproducibility, clinical relevance, and translational potential.

The flowchart presented in Figure 4.4, on the other hand, shows the internal workflow of the Python application for DRF detection. It highlights how user interactions and model outputs are systematically captured, organized, and made accessible for future reference or auditing.

The application begins with the upload of a radiograph, which triggers model inference. Upon completion, the resulting prediction, together with relevant metadata such as confidence score, prediction time, and image identifier, is directed to the Central Logging Module. In parallel, medical professionals may submit feedback at any point—before or after inference, which is similarly accompanied by metadata and passed to the logging component.

The Central Logging Module acts as a coordination unit that channels predictions and

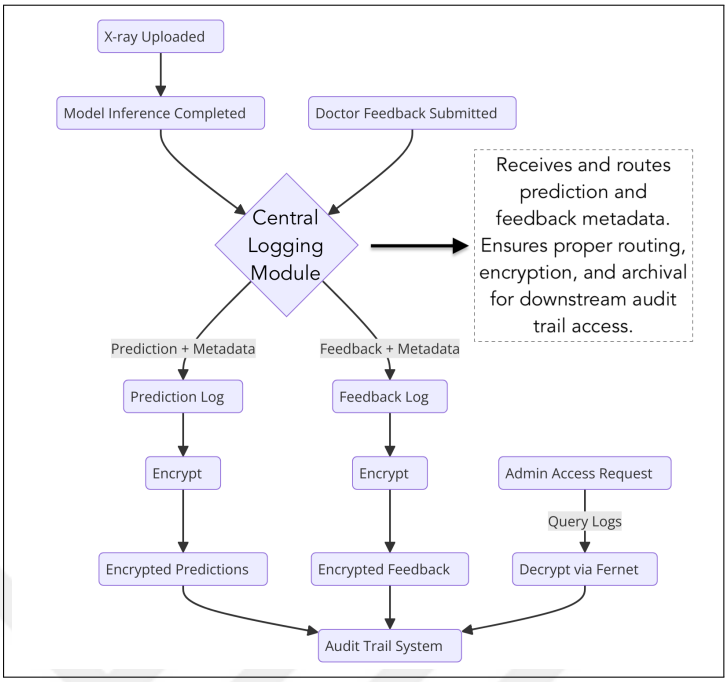


Figure 4.4. The flowchart for the Python-based application.

feedback into two separate streams: the Prediction Log and the Feedback Log. Before being stored, both streams are encrypted to ensure that sensitive information remains protected. All encrypted records are subsequently integrated into the audit trail system for traceability and secure access when required.

4.3. CLASS DIAGRAM FOR THE PYTHON APPLICATION

The class diagram for the Python application is presented in Figure 4.5 and describes the modular design of the application, organized into four primary classes: *PredictionThread*, *PredictionApp*, *Predictor*, and *Decryptor*. Each class encapsulates a specific set of responsibilities.

The *PredictionThread* class represents a threaded component, responsible for handling background operations such as running the prediction process asynchronously. It includes an initializer method and a run method for the thread, enabling non-blocking execution and maintaining responsiveness in the GUI.

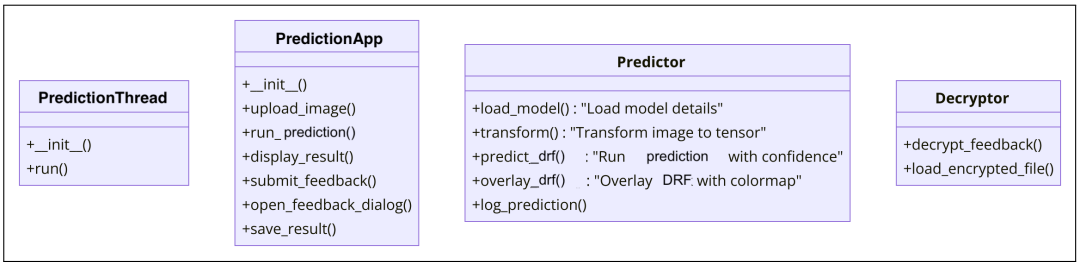


Figure 4.5. The class diagram for the Python-based application.

The *PredictionApp* class serves as the central controller of the application. It manages user interactions and the overall application flow. Key methods include *upload_image*(), *prediction*(), and *display_result*(), which cover the full cycle from loading an image to showing model output. Additional methods like *submit_feedback*() and *open_feedback_dialog*() handle user input and feedback collection, while *save_result*() enables output export. This class interfaces directly with both the GUI components and the underlying prediction and logging modules.

The *Predictor* class encapsulates all model-related operations. It includes methods such as *load_model*(), which loads the model and its parameters, and *transform*(), which preprocesses the input image into a format suitable for inference. The core prediction logic is handled by *predict_drf*(), which returns a confidence score along with the segmented region. Post-processing is handled by *overlay_drf*(), which applies a colormap to the detection for visualization, and *log_prediction*(), which records the prediction metadata.

Finally, the *Decryptor* class manages secure access to stored logs. It includes *decrypt_feedback*() and *load_encrypted_file*(), enabling authorized users (e.g., admins) to access encrypted doctor feedback and model prediction logs through a Fernet-based encryption scheme.

4.4. SEQUENCE DIAGRAM FOR THE PYTHON APPLICATION

This sequence diagram in Figure 4.6 illustrates the runtime behavior of the Python-based DRF detection application, capturing the interactions between the users, the graphical user interface (GUI), the underlying processing thread, the model prediction module, the encryption unit, and the logging system. It provides a clear, step-by-step depiction of how

data flows through the system during image analysis and feedback submission.

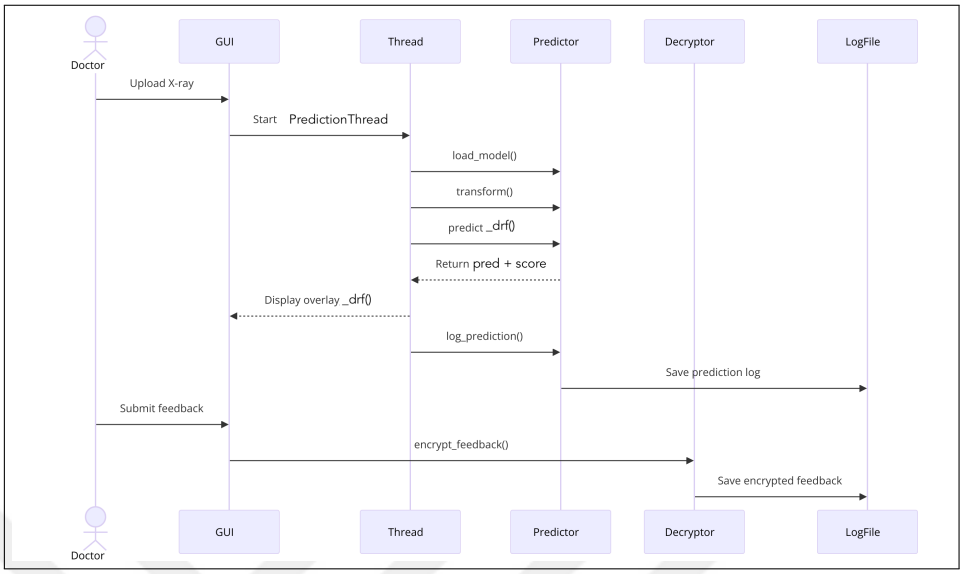


Figure 4.6. The sequence diagram for the Python-based application.

The sequence begins when a doctor uploads an X-ray via the GUI, triggering a background thread $PredictionThread$ to run the model inference without blocking the interface. This thread invokes the $Predictor$ module to load the model using $load_model()$, prepare the input with $transform()$, and generate a prediction with $predict_drf()$, returning a segmentation result and confidence score. The result is displayed on the GUI using $display_overlay_drf()$.

Subsequently, $log_prediction()$ records the prediction and metadata, forwarding them to the logging system. If feedback is submitted, it is handled by the GUI, passed to the $Predictor$, and encrypted using $encrypt_feedback()$. The encrypted output is then stored securely via the $Decryptor$ module in the log file system.

5. DATASET PREPERATION

To train deep learning models for fracture detection and segmentation, a dataset is curated from multiple sources, ensuring that only wrist radiographs with fractures are included. The primary dataset is selected from FracAtlas [97], a publicly available X-ray dataset containing various musculoskeletal radiographs. However, since this dataset includes images from different anatomical regions, it is filtered to retain only wrist radiographs which consist of fractured wrists. The remaining images are then verified and refined by an orthopedic surgeon to ensure the accuracy of the annotations.

In addition to FracAtlas, GRAZPEDWRI-DX [98], another publicly available dataset that specifically focuses on pediatric wrist trauma, is utilized. To maintain data diversity while minimizing redundancy, images are filtered to reduce the repetition of radiographs and added to the dataset. Furthermore, a real-world dataset comprising 43 radiographs from 18 different patients is included, obtained directly from various orthopedic surgeons. These additional images are crucial as they are ensuring the dataset represents real clinical variability in DRFs and increasing the dataset.

All dataset annotations are conducted using Roboflow [67], a tool for image annotation procedure. The annotation format is decided based on the type of model being trained -e.g. for detection models, namely YOLOv8, YOLOv11, and Faster R-CNN, the images are annotated using bounding boxes as presented in Figure 5.1a where the bounding box marks the fractured area in a box. For segmentation models, on the other hand, the bounding box annotations are converted to polygon annotations manually, and then verified by the orthopedic surgeon. As an example, a polygon annotation conversion result is presented in Figure 5.1b.

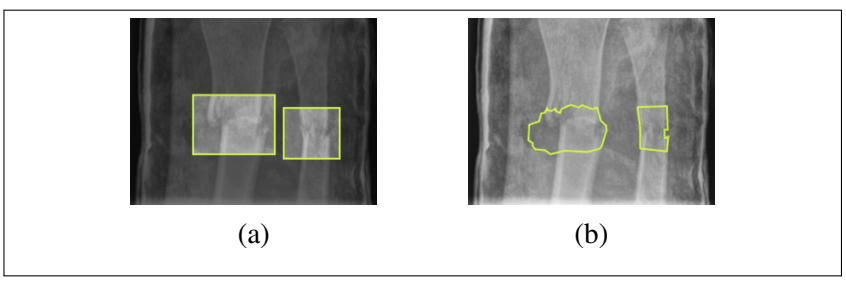


Figure 5.1. (a) Bounding box annotations for detection models, (b) bounding box annotations converted to polygon annotations for segmentation models.

Dataset preprocessing standardizes the data to ensure consistency for the model, as described in Chapter 3. All images in the dataset are resized to 640x640 pixels for both the detection and segmentation datasets. Furthermore, the same augmentations are applied to both datasets to enhance model generalization, which is a particularly important procedure given the limited availability of medical imaging datasets. Table 5.1 presents all of the augmentation operations conducted on the datasets, along with their descriptions.

Table 5.1. Image augmentation techniques

Augmentation Type	**Description**
Flip	Flips the image **horizontally and vertically**.
Rotation (Fixed)	Rotates images by **fixed 90° angles** (clockwise, counter-clockwise, upside-down).
Rotation (Random)	Applies **random small-angle rotations** (-20° to +20°).
Shear Distortion	Applies **shear transformations** (±15° horizontal, ±15° vertical).
Hue Shift	Adjusts image hue **between -20 and +20**.
Saturation Shift	Modifies **saturation by -35% to +35%**.
Exposure Shift	Alters **exposure by -15% to +15%**.

The flip augmentation is applied to handle different anatomical orientations as DRFs might appear on either the left or right side of the image. The fixed rotation augmentation ensures that the model can correctly detect fractures regardless of the image's initial orientation while random rotations add more minor variations to images, preventing the model from over-relying on specific angular positioning. Shear transformations, however, simulate perspective distortions, making the model robust to slight variations in imaging angles. Finally, hue, saturation, and exposure adjustments are employed to introduce variability in contrast and brightness, in order to aid the model perform well across different X-ray machine settings and lighting conditions. These augmentation steps help the dataset become more diverse, reduce the risk of overfitting, and improve the model's robustness in detecting and segmenting fractures accurately.

For the training of all model architectures, the dataset is divided into 70% training, 20% validation, and 10% testing sets. In all of the dataset versions, the same images are maintained in the same respective set to ensure consistency. After these steps, the final dataset consisted of 3,420 X-ray images in total. Finally, the before the processed dataset is used for the training

of various model architectures, it is divided in training, validation, and testing sets, with 70%, 20%, and 10% ratio respectively.

6. IMPLEMENTATION OF DETECTION MODELS

Deep learning models for medical imaging face unique challenges. Unlike traditional classification tasks, fracture detection requires models to localize and outline small, irregularly shaped structures that often blend into the surrounding bone. Given the complexity of fractures that can range from subtle hairline cracks to more apparent breaks, both approaches must be examined to determine their strengths and limitations.

To systematically approach this problem, the implementation is divided into two main parts: detection and segmentation. For the detection approach, three different architectures are trained: YOLOv8 (Medium & Large) [92], YOLOv11 (Medium & Large) [94], and Faster R-CNN [83]. These models focus on detecting fractures using bounding boxes, each with varying trade-offs between speed, accuracy, and localization precision.

Compared to binary classification, object detection is much more meaningul in this context, as it allows models to not only determine the presence of a fracture but also localize its position within an X-ray using bounding boxes. Unlike segmentation models, which provide pixel-wise mask predictions, detection models focus on coarse localization, making them computationally more efficient and well-suited for scenarios where precise contour extraction is not necessary.

To evaluate the effectiveness of object detection models on DRFs, multiple models are trained independently. YOLO-based architectures (YOLOv8 Medium & Large and YOLOv11 Medium & Large) are selected due to their ability to perform real-time detection while maintaining high accuracy. Additionally, Faster R-CNN is included as another type of model, given its reputation for high detection precision in medical imaging applications and it has a different structure compared to YOLO.

These models are trained on the dataset using bounding box annotations, with a structured training pipeline designed to optimize performance. Given that detection accuracy is prioritized over real-time speed, larger YOLO architectures are chosen alongside medium-sized versions to leverage their higher ability for feature extraction and small object detection. While larger models generally come at the cost of slower inference, this trade-off is acceptable in a clinical setting where detection precision is more critical than real-time

processing speed. Accordingly, all models are trained on a Google Colab platform with an A100 GPU with a 40 gigabyte (GB) video random access memory (VRAM) and 83.5GB of system random access memory (RAM).

6.1. TRAINING OF FASTER R-CNN MODEL

For the Faster R-CNN model, the dataset annotations are converted to COCO format. Initially, the necessary libraries and tools such as torch, torchvision, fvcore, and cocoapi are installed.

Algorithm 6.1. Registering COCO Dataset in Detectron2

Require: COCO-formatted dataset with images and annotations
1: Import `MetadataCatalog` and `DatasetCatalog` from Detectron2
2: **Register training dataset:**
3: `register_coco_instances("COCO_train", {}, "train_annotations.json", "train_images/")`
4: **Register test dataset:**
5: `register_coco_instances("COCO_test", {}, "test_annotations.json", "test_images/")`
6: **Register validation dataset:**
7: `register_coco_instances("COCO_val", {}, "val_annotations.json", "val_images/")`
8: **Retrieve Dataset Metadata:**
9: `COCO_train_metadata` ←`MetadataCatalog.get("COCO_train")`
10: `COCO_test_metadata` ←`MetadataCatalog.get("COCO_test")`
11: `COCO_val_metadata` ←`MetadataCatalog.get("COCO_val")`
12: **Load Image Annotations:**
13: `dataset_dicts` ←`DatasetCatalog.get("COCO_train")`
14: `dataset_dicts_test` ←`DatasetCatalog.get("COCO_test")`
15: `dataset_dicts_val` ←`DatasetCatalog.get("COCO_val")`
16: **Return registered datasets for training, validation, and testing.**

Thereafter, the Detectron2, a deep learning framework built on PyTorch for object detection, is utilized to conduct Faster R-CNN training due to version problems as Faster R-CNN is an

old model and its implementation relies on previous versions of PyTorch that is no longer directly supported on Colab.

Since the dataset was initially in COCO format, it had to be registered for use in Detectron2. Algorithm 6.1 shows the steps of this process where necessary Detectron2 modules for dataset registration are imported, training, validation, and testing datasets are registered using `register_coco_instances()` function that specifies the annotation files and image directories, and the dataset annotations are loaded to match the bounding boxes with the corresponding images. Lastly, the final step ensures that the datasets are registered correctly and ready for training.

A new list of configurations is created for the training, which begins by calling get_cfg(). This is followed by merging a pre-existing Faster R-CNN configuration file to define the model architecture. A Faster R-CNN with a ResNet-101 backbone and an FPN is selected due to its ability to enhance feature extraction for small object detection, provide multi-scale feature representation via FPN, and improve localization accuracy compared to ResNet-50. To further accelerate training and improve generalization, pre-trained weights from a Faster R-CNN model trained on COCO are loaded, implementing transfer learning to leverage prior knowledge from large-scale object detection datasets.

As for hyperparameter tuning, the number of CPU threads used for data preprocessing is set to 2, optimizing data loading speed. The batch size per iteration is set to 4, which provides a balance between memory efficiency and training stability. A low base learning rate of 0.0001 is chosen, which is especially vital for fine-tuning pre-trained models on new datasets to prevent drastic weight updates that may lead to instability. The training process ran for 8,000 iterations, with a scheduled learning rate reduction at iterations 5,000 and 7,500, where the learning rate is decreased by a factor of 10. This learning rate decay at these specific steps helps prevent the model from converging too quickly, consequently getting stuck in a local minimum. Furthermore, this refines weight updates in later training stages for improved performance.

With the goal of evaluating the model during and after training, a custom evaluator is constructed based on the COCOEvaluator provided by Detectron2 that computes key metrics such as mean average precision, recall, and the F1-Score. Every 100 iterations; validation is

conducted on the model to track the improvements over time.

Additionally, the RoI head parameters are also adjusted to optimize detection performance. These parameters are responsible for classifying detected objects and refining bounding box predictions based on region proposals. The batch size per image is specified to be 256, which defines how many region proposals are sampled per image. A larger batch size allows the model to evaluate more potential fracture regions, which is particularly beneficial for detecting small fractures that may otherwise be overlooked. For efficiency, the number of classes is also defined as 2. The first class is ”fracture” whereas the other class is ”background”. The 8000 iterations took a total of 35 minutes over 8000 iterations, approximately 0.2 seconds per iteration.

As efficient data loading is crucial for optimizing GPU utilization and maintaining steady training progress, the data loading time (data_time) and ranked data loading time (rank_data_time) graphs presented in Figure 6.1a and Figure 6.1b respectively. The graphs demonstrate how training efficiency evolved over iterations. Initially, data transfer overhead was high (0.8s per iteration), but as training progressed, the system stabilized, achieving near-zero data loading delays. After 1000 iterations, the time stabilizes close to zero, indicating that data is efficiently loaded into memory and training is no longer bottlenecked by input/output operations. Possibly due to dataset caching and CPU-GPU memory transfer overhead, data loading time in Figure 6.1b initially starts with high values. After the first 1000 iterations, the time drops sharply, stabilizing at a value close to 0, which indicates that data preprocessing and loading became more efficient as training progressed. Similarly, the estimated time remaining per iteration (eta_seconds) graph shown in Figure 6.1c exhibits a sharp decline, indicating that training initially progressed slowly due to weight adjustments, however, it stabilized as the optimizer converged. By the end of the training, the estimated time of arrival (ETA) reaches 0, meaning the training process has been completed. The steady reduction in the ETA reinforces that the training pipeline has become increasingly efficient.

The total loss (total_loss) graph illustrated in Figure 6.2a accumulates the overall loss contributions from region proposals, classification, and bounding box refinement. Unlike an ideal training curve that smoothly decreases, this graph exhibits significant fluctuations between the points 0.40 and 0.44, suggesting that the model encountered variability

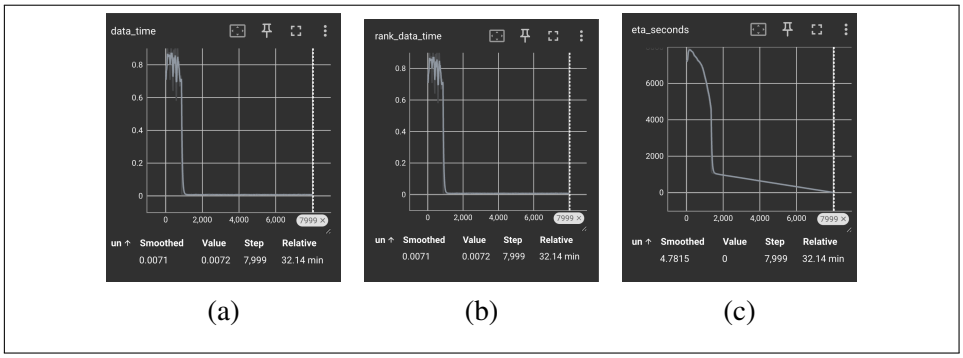


Figure 6.1. (a) Time taken for data loading per iteration, (b) data loading time during training, (c) the estimated time remaining for training.

in gradient updates. This instability could stem from either bounding box regression inconsistencies, as indicated in Figure 6.2b, which shows fluctuating values instead of a steady decrease, or the RPN localization challenges, presented in Figure 6.2c, which demonstrates a more stable downward trend but still exhibits noise or fluctuations. In any case, while the RPN localization loss depicted in Figure 6.2c indicates improving region proposals, the bounding box regression loss remains inconsistent, which might imply that the network struggles with precise fracture localization, possibly due to small object size.

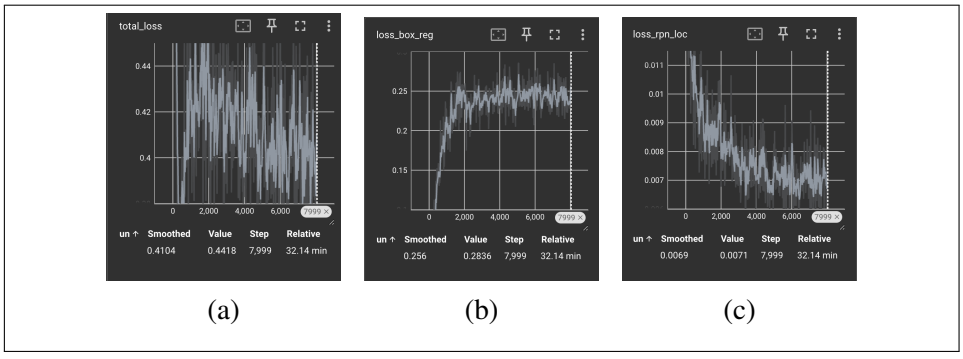


Figure 6.2. (a) Total loss, (b) bounding box regression loss, (c) RPN localization loss for Faster R-CNN training.

More specifically, Figure 6.2b measures how well the model adjusts bounding box coordinates to match ground truth fractures and the final loss value is 0.28. Figure 6.2c, on the other hand, represents the localization loss for the RPN, which measures how well the generated anchor boxes align with the ground truth fractures. Initially, the loss value is high, however, the loss steadily decreases through training, stabilizing around 0.007 after all the iterations.

Faster R-CNN's performance heavily depends on the RPN, which identifies potential

fracture locations before classification. The RPN positive and negative anchor assignments provide insight into how the model selects bounding boxes for refinement. The $rpn/num_pos_anchors$ graph presented in Figure 6.3 shows a low number of positive anchors, approximately 4.6 per iteration, which means that a low number of proposals are correctly matched with fractures. This is expected though as fractures are small and sparse. Conversely, the $rpn/num_neg_anchors$ graph remains stable around 251 negative anchors, indicating that most region proposals are rejected as background.

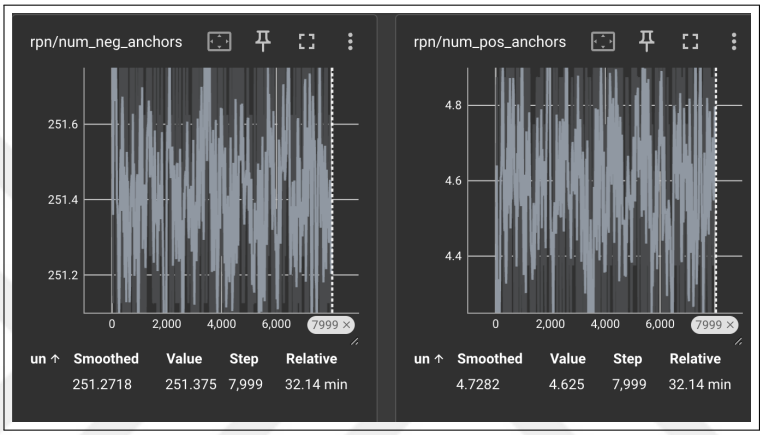


Figure 6.3. Number of positive and negative anchors by the RPN.

A similar trend is observed in the RoI sampling such that the $roi_/num_fg_samplesgraph$ presented in Figure 6.4 shows an increase in foreground, which is fracture class, samples over training, implying that the model gradually learns to detect more fractures. The $roi_/num_bg_samples$ graph, on the other hand, shows a decline in background samples, supporting that the detector becomes more confident in distinguishing fractures from irrelevant regions. Thus, these patterns suggest that Faster R-CNN effectively learns to localize fractures over time.

6.2. TRAINING OF YOLO ARCHITECTURE BASED MODELS

The YOLOv8m and YOLOv8l models are trained for DRF detection using the same dataset and training environment as Faster R-CNN to ensure consistency. Both models are trained using Ultralytics' YOLOv8 framework and executed on an A100 GPU in a Colab environment. Only the annotation format of the dataset is altered to YOLO format and the standard image resolution of 640x640 is maintained for both of the trainings. The dataset

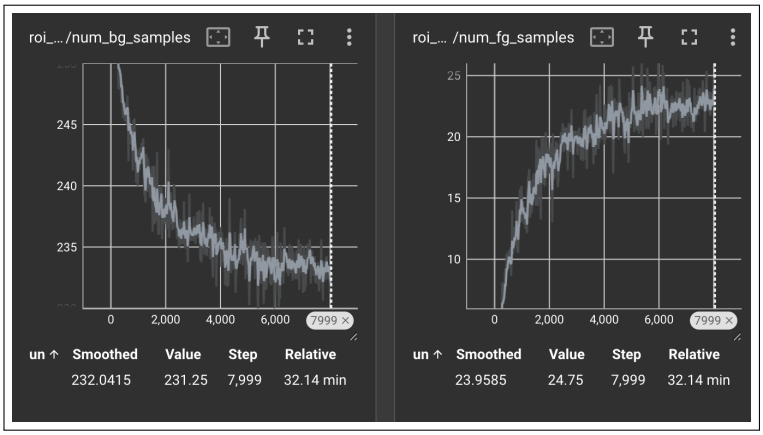


Figure 6.4. Number of background and foreground sample by the RoI.

is split into three sets, namely 70% training, 20% validation, and 10% testing since. YOLO utilizes validation data for hyperparameter tuning during the training process. The same augmentations as Faster R-CNN are applied to the dataset, including horizontal and vertical flipping, rotations, shear transformations, and exposure adjustments.

The YOLOv8m model is initialized from pre-trained COCO weights and fine-tuned by altering various hyperparameters, namely batch size, epoch number, optimizer, learning rate schedule, IoU threshold, maximum detection number, and warm-up epochs. The batch size is set to be 8, which means that the model processes 16 images prior to updating its weights. While larger batch sizes speed up convergence, they commonly require higher learning rates to compensate. Smaller batch sizes, however, might increase noise in gradient updates. 16 images per batch ensure a balance between stability and generalization. An epoch is a full pass through the entire dataset and 100 epochs mean that the model sees the entire dataset 100 times while refining its parameters iteratively. The training is conducted with 50 epochs initially, however, after observing that the model does not overfit and has room for improvement, it was trained again for 80 epochs. Once more, the model still had some room for improvement before overfitting occurs and since small objects like fractures require longer training to improve localization accuracy, the model is trained for 100 epochs, with the metrics finally stabilizing, indicating the epoch count of 100 is enough.

The default optimizer, Adam Weight Decay (AdamW) with a learning rate of 0.002 and momentum of 0.9 is utilized to adjust the model weight and minimize loss. AdamW is an improved version of Adam such that it prevents overfitting by regularizing weight updates

by penalizing large updates. The momentum, on the other hand, helps accelerate gradient descent by preventing oscillations and improving convergence speed. The ioU parameter is crucial as it measures how well a predicted bounding box aligns with the ground truth to be considered valid. In this training, the optimal value is decided by exploration. The best-performing value turned out to be 0.6, as lower thresholds increase false positives, despite improving recall.

The number of maximum bounding boxes allowed per image is set to 200, since the model is required to find fewer objects from radiographs compared to general object detection datasets; reducing the computation time. Lastly, the warm-up epoch gradually increases the learning rate to prevent unstable updates at the start of the training, which was observed with Faster R-CNN. For the first 3 epochs, the learning rate starts lower and momentum is set to 0.7 to avoid rapid early updates, stabilizing the training. Overall, the training process took 0.646 hours for 100 epochs.

The YOLOv8l model is also trained under the same hyperparameter settings. The YOLOv8l model, however, has more convolutional layers, more parameters, and higher FLOPs, requiring more computational cost and memory. Thus, the training took more time, a total of 0.986 hours.

Figure 6.5 shows the bounding box distribution of dataset by illustrating the spatial distribution of bounding boxes across the datasets. It is generated after the model training and is identical in both medium and large models, as the provided annotations are the same. The top-left solid blue box represents the total number of labeled instances, indicating that all annotations belong to a single fracture class as classification is not conducted. The top-right section shows an overlay of all bounding boxes projected onto a normalized image space. The dense clustering indicates that most fractures are located centrally, likely corresponding to common wrist fracture sites. This claim is backed by the bottom-left scatter plot of x and y coordinates, which further confirms that most annotations are concentrated in the center, with some distributed throughout the radiographs due to augmentations. Lastly, the bottom-right width-height scatter plot shows that most bounding boxes are relatively small, which is a recognized challenge in fracture detection. These plots provide insights into how fractures are acknowledged by the models.

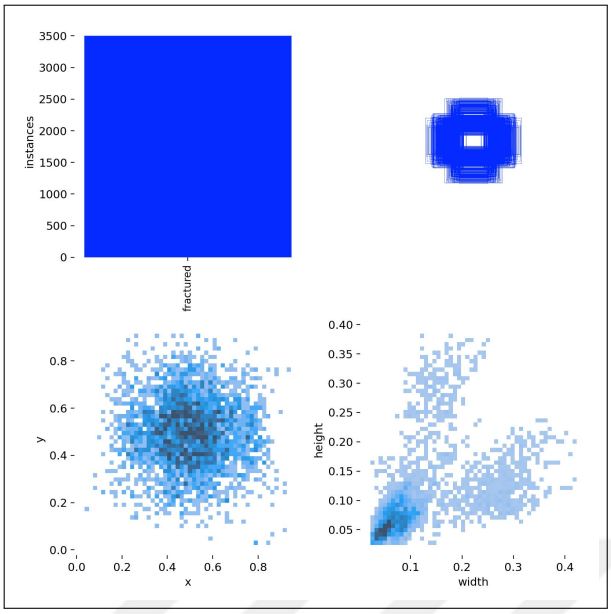


Figure 6.5. Bounding box distribution of the dataset, produced by the trained YOLOv8 models.

Figure 6.6, on the other hand, analyzes the relationships between different bounding box attributes, providing deeper insights into size distributions, spatial locations, and correlations. The diagonal histograms show the distribution of each attribute, x, y, width, height, across the dataset. The x–y scatter plot shows that although fractures are distributed throughout the image space, they predominantly cluster near the center, indicating that most fractures occur in anatomically central regions of the radiograph.

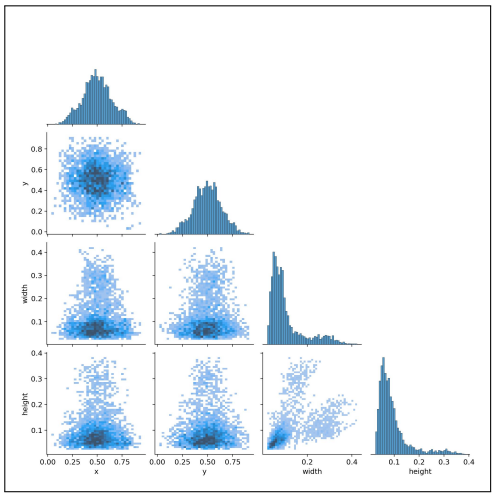


Figure 6.6. Bounding box attributes of the dataset, produced by the trained YOLOv8 models.

The YOLOv11m and YOLOv11l models are trained under the exact same conditions as the previously trained YOLOv8 models to ensure non-biased comparability. The same dataset, preprocessing steps, augmentation strategies, and training environment are maintained to eliminate external variations that could influence performance differences between model architectures. Both models are trained with the same hyperparameters as YOLOv8 models, including batch size, learning rate, optimizer, and IoU threshold, allowing for direct evaluation of improvements in YOLOv11 over YOLOv8. The training of the YOLOv11m model took a total of 0.736 hours whereas the YOLOv11l model took 1.11 hours. Table 6.1 shows a summary of the adjusted parameters for both the YOLOv8 and YOLOv11 detection models.

Table 6.1. Hyperparameters for YOLO detection models

Hyperparameter	**Value**
Training Task	Object Detection
Input Image Size	640 × 640
Batch Size	16
Epochs	100
Optimizer	AdamW (with lr=0.002, momentum=0.9)
Learning Rate Schedule	Linear Decay
Warm-up Strategy	3 epochs, momentum = 0.7
IoU Threshold	0.6
Max Detections per Image	200
Augmentations Applied	Horizontal & Vertical Flip, Rotation (-20° to +20°), Shear (±15°), Hue (-20° to +20°), Exposure (-15% to +15%)
Dataset Split	70% Train / 20% Validation / 10% Test

7. IMPLEMENTATION OF SEGMENTATION MODELS

To achieve segmentation, four model architectures are trained separately: YOLOv8 (Medium & Large) segmentation models [92], YOLOv11 (Medium & Large) segmentation models [94], Mask R-CNN [70], and EfficientNet [80]. Unlike detection models, which localize fractures within a bounding box, segmentation models generate pixel-wise masks, offering a more detailed representation of fracture regions. The goal is to evaluate whether these models can effectively highlight fracture boundaries while handling challenges such as small object segmentation and variations in fracture presentation. Thus, the following sections detail the dataset preparation, training configurations, optimization strategies, and key observations from training each model.

The same environment mentioned in Chapter 6 is employed to ensure consistency. Models are trained on a Google Colab environment with an A100 GPU with a 40 GB video VRAM and 83.5GB of system RAM. The utilized dataset is also identical, only difference being the annotation format as instance segmentation models require pixel-wise mask annotations.

7.1. TRAINING OF MASK R-CNN MODEL

First segmentation model that uses Mask R-CNN architecture is trained using Detectron2. For the backbone of the Mask R-CNN model, ResNet101 with FPN is preferred to enhance multi-scale feature representation. While the Res-Net architecture provides the extraction of multi-scale hierarchical features from input radiographs, the FPN lateral and output layers deliver the progressive refinement of spatial information across different scales. For the RoI head of the ResNet architecture, the box head predicts the bounding boxes around the detected fractures while the mask head outputs pixel-wise masks for each region of interest. Furthermore, RoIAlign is utilized to ensure accurate mask prediction by aligning feature maps with proposal regions.

The hyperparameters used in Mask R-CNN training are carefully selected to balance segmentation accuracy and computational efficiency. Table 7.1 presents the adjusted hyperparameters and their respective values.

Table 7.1. Mask R-CNN model hyperparameter adjustment

Hyperparameter	**Value**
Batch Size	2
Base Learning Rate	0.0001
Max Iterations	8000
Step Learning Rate Decay	[5000, 7000]
Learning Rate Decay Factor	0.2
ROI Batch Size per Image	128
Number of Classes	2
Mask Head Activation	True

Compared to Faster R-CNN model training, there are some modifications conducted to ensure the resulting model performs well as the models have similar structures, however, different prediction procedures. The batch size is reduced compared to the Faster R-CNN training, set at 2, as segmentation tasks require more memory due to pixel-wise mask predictions. Each mask prediction adds additional computational overhead, making it necessary to use a smaller batch size to prevent out-of-memory (OOM) errors on the GPU. The small learning rate is maintained to allow fine-tuning without drastic weight updates. Since Mask R-CNN is initialized with a pre-trained model, a low learning rate ensures gradual adaptation to the wrist fracture dataset.

Identical to Faster R-CNN training, the number of iterations is set to be 8000 where the learning rate at iterations 5000 through 7000 is lowered by 0.2 to allow finer adjustments in the later training stage. The decay in learning rate is expected provide improvement in segmentation mask accuracy over time. Similarly, the RoI batch size is reduced to 128, to define how many region proposals are sampled per image for training. The lower value in this instance ensures memory overloading does not occur. Lastly, the mask head activation value is set to true to ensure that Mask R-CNN produces pixel-wise segmentation masks instead of only bounding boxes. Once more, the AdamW optimizer is employed to handle small-scale object learning better. The training took a total of 66 minutes, approximately 0.49 seconds per iteration, excluding the inference times on the validation set. Furthermore, an additional class is implemented to ensure the best performing step throughout the 8.000 iterations is saved as a checkpoint in terms of evaluation metrics. This model will be utilized for testing later on.

Figure 7.1 shows the details regarding the RPN where the left graph presents the number of positive anchors and the left graph delivers the number of negative anchors. The number of negative anchors remains stable at around 251.5 throughout the training, indicating that the RPN is maintaining a consistent balance between accepting and rejecting candidate proposals. The number of positive anchors, on the other hand, fluctuates around 4.5, suggesting that only a small fraction of region proposals are being assigned as foreground. This reflects the difficulty of detecting small fractures, as only a few proposals have sufficient overlap with the actual fracture regions to be considered relevant. In conclusion, the small number of positive anchors relative to negative ones suggests that fractures are sparse and small. The RPN is learning to filter out irrelevant regions effectively, but it must ensure that sufficient positive samples are retained to improve detection.

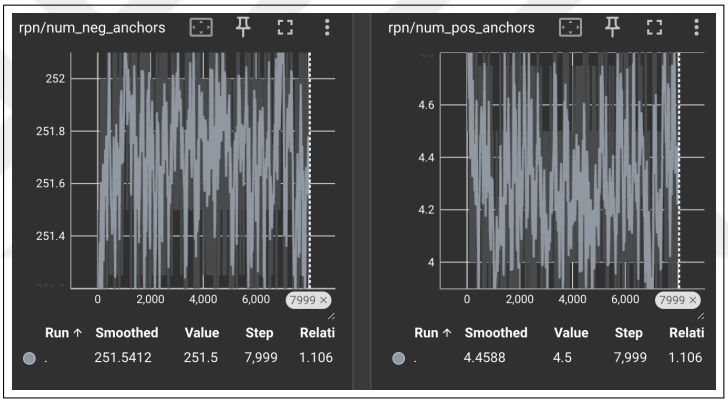


Figure 7.1. The number of positive and negative anchors proposed by the ROI.

The false negatives produced by the model during training start at a relatively high level, as depicted in Figure 7.2a. A false negative occurs when the model fails to detect a fracture that is actually present in the image, leading to a missed diagnosis. Over the course of training, the false negative loss gradually decreases and stabilizes around 0.1, suggesting that the model has learned to identify fractures more reliably. However, despite this stabilization, some fractures may still go undetected, which could affect the model's overall sensitivity. While additional training epochs could have been conducted to further refine detection performance and ensure that this false negative rate remained consistently low, this approach was not pursued due to concerns about overfitting. Extending training beyond this point could have led the model to memorize specific patterns in the dataset rather than generalizing well to unseen cases.

However, the false positive rate, which represents instances in which the model incorrectly

predicts a fracture where none exists, exhibits a substantial decline throughout training, as illustrated in Figure 7.2b. Initially, the model produces a large number of false positives, likely due to the complexity of the bone structures and the similar appearance of normal anatomical variations to fractures. However, as training progresses, the false positive rate gradually drops, reaching approximately 0.2 by the end of training. This steady decline indicates an improvement in the model's precision, meaning that it is becoming more selective in its fracture predictions and reducing unnecessary alerts. A lower false positive rate is particularly beneficial in a medical setting, as excessive false alarms can lead to unnecessary follow-up examinations, increased workload for radiologists, and potential patient anxiety.

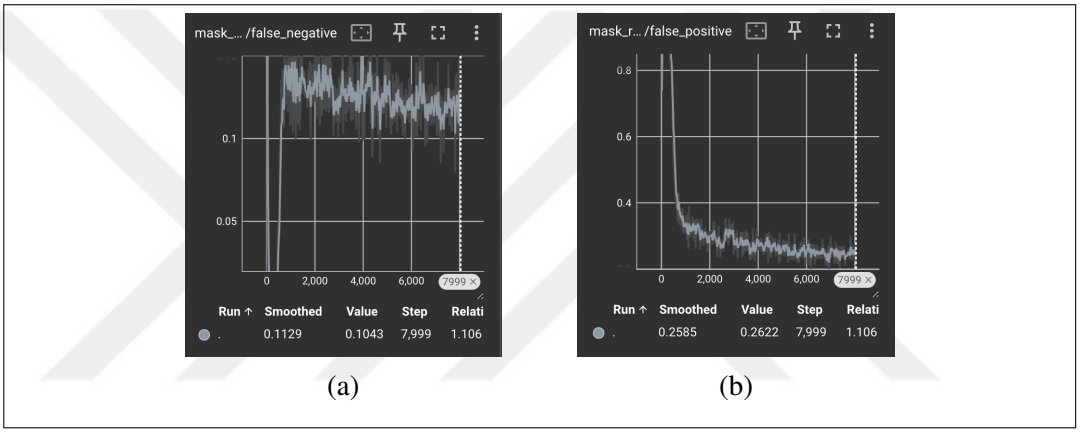


Figure 7.2. (a) Mask R-CNN false negatives, (b) false positives during training.

Lastly, the total loss graph shown in Figure 7.3 starts high and steadily decreases by the end of training. The decreasing trend suggests a successful convergence, which means that the model is improving in both localization and mask prediction for fractures. The overall downward trend is desirable despite some fluctuations.

7.2. TRAINING OF EFFICIENTNET MODEL

The first step undertaken to prepare the dataset for training the EfficientNet segmentation model is loading the COCO-formatted annotations. The dataset follows the widely used COCO annotation format, which stores image metadata along with segmentation annotations as polygon coordinates. To access this information, a script utilizes the pycocotools.coco library, loading the annotation file that contains image filenames, sizes, and corresponding segmentation masks. This process ensures that all annotations are properly retrieved before

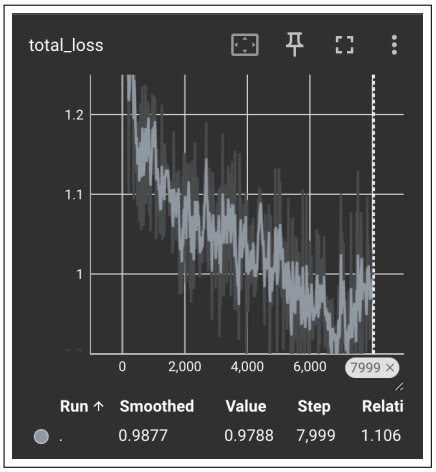


Figure 7.3. The total loss graph of Mask R-CNN model training.

training begins. Once the annotations are loaded, the script selects the first image from the dataset and retrieves its metadata, including its dimensions and file name. The image is then loaded using OpenCV, and since OpenCV reads images in the BGR format by default, it is converted to the RGB color space to ensure correct visualization when plotted using Matplotlib. This step helps verify that the dataset paths are correctly structured and that the images are being accessed properly. After loading the image, the script constructs its corresponding segmentation mask. This is achieved by extracting the polygon-based segmentation data from the COCO annotations. Since each segmentation mask consists of a series of coordinate points outlining the fracture, the script initializes an empty mask of the same dimensions as the image and fills the regions defined by these polygons with a value of 1, effectively creating a binary mask. Figure 7.4 shows the output of the script, where the initial X-Ray from the dataset is shown on the left side, and the corresponding segmentation mask is presented on the right.

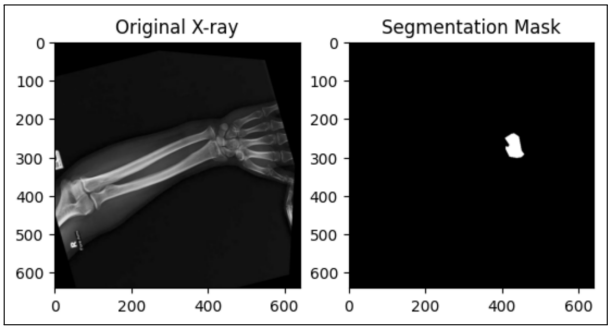


Figure 7.4. A prepared image from the dataset for EfficientNet training.

The next phase of training involves constructing the EfficientNet model architecture. The EfficientNet-B4 encoder is utilized with a U-Net backbone. The choice of EfficientNet-B4 balances model complexity and performance, offering strong feature extraction capabilities without an excessive computational burden. The model utilizes pre-trained weights from the ImageNet dataset, which aids in accelerating the convergence by leveraging features learned from large-scale image classification tasks.

The EfficientNet-B4 encoder involves a stem convolutional layer, which is the first layer in the encoder to perform the padding operation. Furthermore, it performs batch normalization to normalize activations, stabilizing training and improving generalization. It also employs mobile inverted bottleneck convolutional blocks that operate at different feature scales to progressively refine feature representations while reducing spatial dimensions.

The U-Net decoding path, on the other hand, reconstructs the segmentation masks from the extracted features through the encoder, using up-sampling and convolutional operations. Moreover, attention mechanisms are employed to enhance segmentation accuracy by emphasizing important regions in the feature maps. Each decoder block progressively increases the spatial resolution of the feature maps, refining the output segmentation mask. Lastly, the final component is the segmentation head, which is responsible for producing the final pixel-wise fracture segmentation mask. It consists of two main components, namely the final convolutional layer and the activation function. The final convolutional layer employed is a 3x3 convolution layer with a single output channel for binary segmentation. The activation function, on the other hand, is later replaced with Sigmoid during training to ensure that the output values remain unscaled before thresholding.

The input to the model is assembled and it consists of three channel images, corresponding to RGB input. The output of the model is a single channel segmentation mask where each pixel is classified as either belonging to a fracture or not, making this a binary segmentation problem.

Once the images, masks, and model are prepared, optional augmentations are applied using Albumentations, ensuring that both images and masks undergo identical transformations to maintain spatial consistency. After transformations, the image dimensions are reformatted to match PyTorch's expected input structure by converting them from (H, W, C) to (C, H, W), and

masks are reshaped from (H, W) to (1, H, W) to facilitate binary segmentation. The dataset is then loaded using PyTorch's DataLoader, which batches the data, shuffles it to introduce randomness into the training process, and optimizes memory allocation for GPU training.

The hyperparameters are fine-tuned next. The BCEWithLogitsLoss function is chosen as the loss measure, as it is well-suited for binary segmentation tasks by combining Sigmoid activation with binary cross-entropy. This choice ensures that the predicted segmentation maps produce smooth probability distributions. The Adam optimizer is selected for training, with a learning rate of 0.0001, balancing convergence speed and stability. Adam's adaptive nature allows it to efficiently handle sparse gradients, which is common in medical image segmentation where the majority of the image is background.

The training loop is executed for 25 epochs, where the model iterates through batches of images and masks. Each batch is transferred to the GPU, and predictions are made using a forward pass. The predicted segmentation maps are then compared to the ground truth masks, and the loss is computed. Backpropagation is used to calculate gradients, which are then updated using the optimizer. After every epoch, the average loss is printed to track model performance and detect potential issues such as overfitting or slow convergence. The reduced batch size of 8 is specifically chosen due to the high memory consumption of segmentation models, where each mask is dense and requires large computational resources.

Upon observing that loss values continued to decrease and evaluation metrics such as accuracy and recall kept improving, training was extended for an additional 25 epochs, reaching a total of 125 epochs. To ensure that the model does not overfit during this extended training phase, checkpoints were taken every 5 epochs. These checkpoints were stored so that the best-performing model could be restored in case overfitting was detected before the full training cycle was completed. However, this concern did not materialize, as the model continued to improve throughout the extended training. The checkpointing strategy ensured that a well-generalized model was obtained without unnecessary training that could lead to diminishing returns.

Figure 7.5 illustrates the training loss values over the 25 epochs of the training. Initially, the BCE loss starts at approximately 0.16, which rapidly declines within the first few epochs, demonstrating a sharp improvement in the model's ability to learn meaningful features. By

epoch 5, the loss stabilizes around 0.01, and the subsequent epochs exhibit a more gradual reduction.

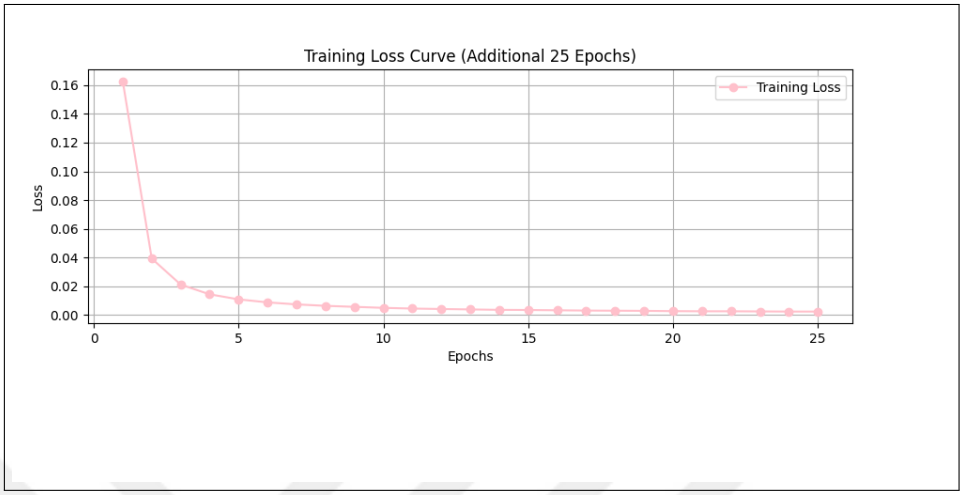


Figure 7.5. EfficientNet loss values during the training.

From epoch 10 onward, the loss values consistently remain low, converging around 0.0024 by the final epoch. This consistent decline indicates that the model continued to improve without significant fluctuations, proposing effective learning. The lack of sudden spikes or instability suggests that the optimizer and learning rate selection are well-tuned. This smooth downward trajectory reinforces that the additional training was beneficial, enhancing the model's segmentation performance without leading to overfitting.

Finally, the training took a total time of 3.4 hours which is comparatively longer than the training of previous models. This can be attributed to several factors. First of all, the SE blocks and depth-wise convolutions in EfficientNet introduce additional computational overhead. Furthermore, unlike object detection models that predict a few bounding boxes per image, segmentation models predict an entire segmentation mask at a pixel level. This means that each pixel in an image must be classified, increasing the number of computations. Additionally, the decoder must reconstruct a high-resolution segmentation mask from deep feature maps, which is computationally demanding.

7.3. TRAINING OF YOLO ARCHITECTURE BASED SEGMENTATION MODELS

The training of YOLOv8m and YOLOv8l segmentation models follows a well structured approach to ensure consistency with previous models while leveraging YOLOv8's real-time segmentation capabilities. Both models are trained on the same dataset used for prior segmentation models, with images resized to 640×640 pixels, however, YOLOv8 annotation format is utilized. To maintain comparability, the training environment is set to be identical to the one used for Mask R-CNN and EfficientNet segmentation models. The models are trained on an A100 GPU in Colab environment. The dataset is split into 70% training, 20% validation, and 10% testing, ensuring the validation set is utilized during training for hyperparameter tuning.

Both the YOLOv8m and YOLOv8l segmentation models are trained using the hyperparameters presented in Table 7.2. The initial learning rate (lr0) is set to a lower value compared to the default value for the beginning of the training to ensure a stable conversion. The final learning rate factor (lrf) controls the decay of the learning rate over training epochs. Momentum, on the other hand, aids in stabilizing updates in gradient descent, consequently preventing abrupt changes in learning. The number of warm-up epochs is incremented to 4, which improves the stability of the initial training phase. Weight decay is a regularization term to prevent overfitting.

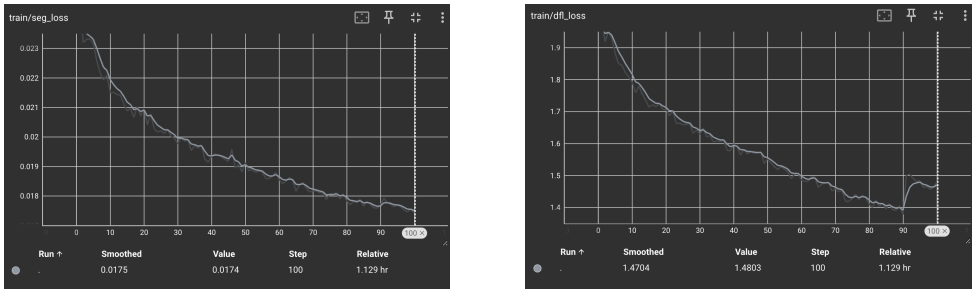
Table 7.2. YOLOv8 segmentation hyperparameters

Hyperparameter	**Value**
Epochs	100
Image Size	640x640
Initial Learning Rate (lr0)	0.0005
Final Learning Rate Factor (lrf)	0.7
Momentum	0.95
Warm-up Epochs	4
Box Loss Coefficient (box)	0.05
Classification Loss Coefficient (cls)	0.8
Weight Decay	0.0005
IoU Threshold	0.7
Max Detections	300
Batch Size	16
Optimizer	AdamW

The loss values, namely box loss coefficient (box) and classification loss coefficient (cls) are likewise adjusted to ensure better training outcomes. While the box loss coefficient penalizes incorrect bounding box predictions to refine spatial segmentations, the classification loss coefficient aids in emphasizing the class confidence predictions. The optimizer, IoU threshold, maximum detection number, and batch size are maintained as in the previous YOLO training.

The models' performances are monitored via mAP@50 and mAP@50:95, tracking the intersection over union (IoU) between predicted masks and ground truth annotations. Additionally, graphs such as loss curves, precision-recall curves, F1-Score across confidence curves, confusion matrices, and mask predictions are also monitored throughout the training.

Training loss graphs for the YOLOv8m model are shown in Figure 7.6 where Figure 7.6a depicts the segmentation loss and Figure 7.6b illustrates the DFL loss. Both of the loss graphs exhibit a consistent downward trend which indicates that the model is effectively learning to optimize both its segmentation outputs and its distribution-based localization throughout the 100 epochs. The segmentation loss starts relatively high and steadily decreases, with no apparent increase towards the end, suggesting that the model was still fitting well to the training data without significant overfitting. Similarly, the DFL loss shows a smooth decline, confirming that the model was refining its object localization and segmentation confidence.



(a) (b)

Figure 7.6. (a) YOLOv8m training segmentation loss, (b) training DFL loss.

On the other hand, the validation losses of the training tell a different story, as presented in Figure 7.7 where Figure 7.7a shows the validation segmentation loss and Figure 7.7b exhibits the validation DFL loss. The validation loss graphs provide insights into how the model generalizes to unseen data. While the validation segmentation loss graph follows a

downward trend initially, mirroring the training loss, after a certain number of epochs, it begins to increase again, which is a strong indicator of overfitting that happens when the model starts to memorize the training data rather than learning the generalizable patterns. Ideally, training should be stopped at the point where validation loss is at its lowest before it begins to rise. Similarly, the validation DFL loss shows high fluctuations throughout training, unlike the smoothly decreasing training DFL loss.

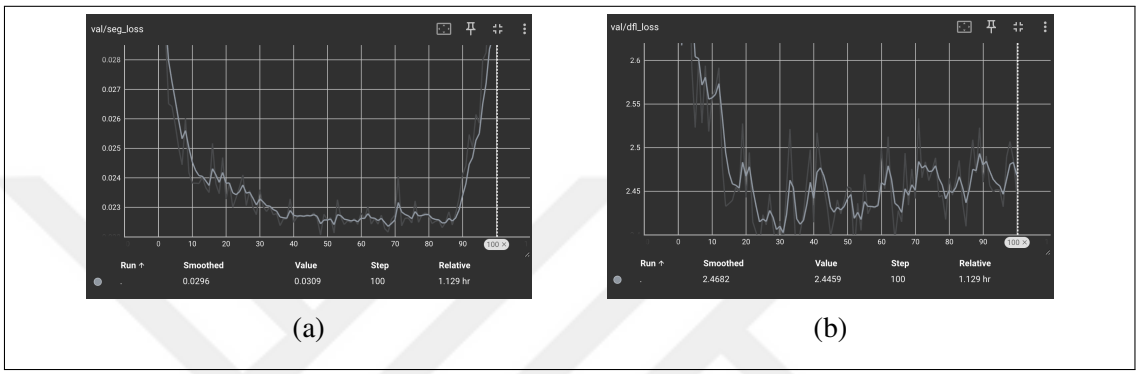
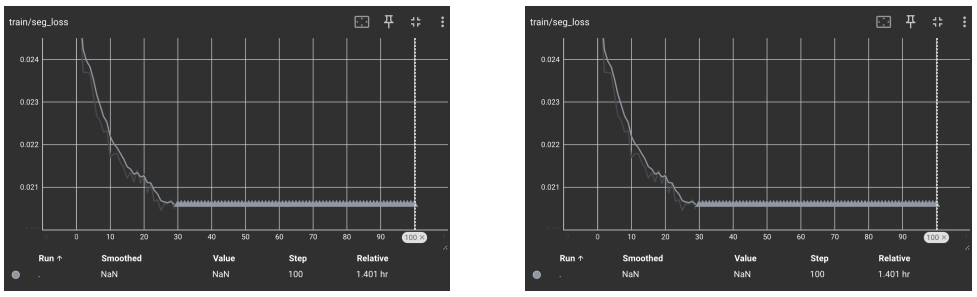


Figure 7.7. (a) YOLOv8m validation segmentation loss, (b) validation DFL loss.

Given these observations, a checkpoint model is saved at the best-performing epoch where the loss values are the lowest and the evaluation metrics are the highest. This approach ensures that the model utilized for later testing is the most optimal in terms of generalization, rather than using the final trained model that is likely to be overfitted. In the testing phase, the best checkpoint will be utilized to evaluate how well the model performs on completely unseen external test data.

The YOLOv8l segmentation model is trained using the exact same hyperparameters, as well as an identical dataset with the same augmentations and annotations. Figure 7.8 shows the segmentation and DFL losses for the YOLOv8l model respectively. Similar to the medium model training, both losses followed a downward trajectory initially, signifying stable learning. However, at approximately epoch 30, both losses plateaued, and from epoch 40 onward, values appear to stagnate completely, showing no further improvements.

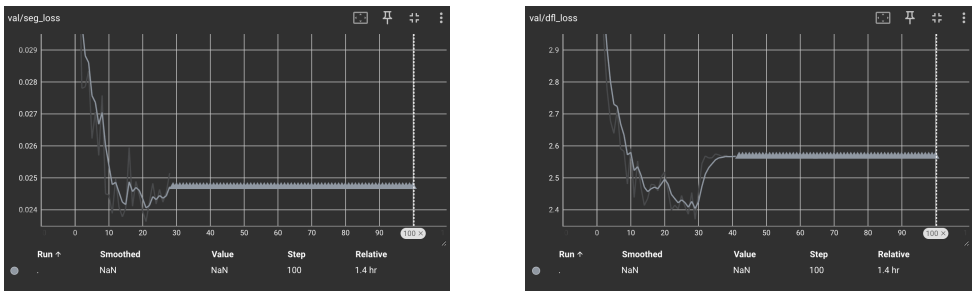
This suggests that training encountered numerical instability, potentially leading to the loss becoming not a number (NaN). The NaN values appearing towards the later epochs indicate that the optimization process likely diverged due to an unstable gradient update, inappropriate learning rate scheduling, or an issue with model weight updates.



(a) (b)

Figure 7.8. (a) YOLOv8l training segmentation loss, (b) training DFL loss.

The validation loss graph for this model during training is presented in Figure 7.9. Similar to the training loss graphs, initially, both the segmentation and DFL losses decline, which implies that the model is effectively generalizing on the validation set. Nonetheless, after approximately 30 epochs, validation losses become erratic, and from epoch 40 onward, they remain flat, showing no further reduction in validation performance. The segmentation validation loss stabilizes at a marginally higher value than the lowest observed loss, whereas the validation DFL loss fluctuates and slightly increases.



(a) (b)

Figure 7.9. (a) YOLOv8l validation segmentation loss, (b) validation DFL loss.

This behavior suggests that the model reached optimal performance around epoch 30 to 40, after which additional training did not yield further improvements. Instead, the increasing fluctuations in validation losses indicate possible overfitting. Consequently, a checkpoint was saved at the best-performing epoch to ensure that the model used for testing is the most generalized version. This saved checkpoint will be used for the final model evaluation on the test set, ensuring that an overfitted version is not inadvertently selected for real-world performance assessment.

The YOLOv8m segmentation model training took a total of 1.143 hours whereas the YOLOv8l segmentation model training took a total of 1.424 hours, due to the increased number of parameters of the YOLOv8l model.

The training process for the YOLOv11m and YOLOv11l segmentation models follows the same structured approach as the previous models to maintain comparability. The dataset remains unchanged, ensuring that the same set of distal radius fracture annotations is used. Furthermore, the data augmentations applied prior to training are identical to those in prior experiments.

To ensure a fair comparison with the YOLOv8 segmentation models, the hyperparameters remain largely unchanged. The initial learning rate is set to 0.0005 once more, with a final learning rate factor of 0.7, ensuring a smooth decay over time. The momentum parameter is 0.95, with a 4-epoch warm-up phase, allowing the optimizer to stabilize before making aggressive updates. The weight decay is set at 0.0005, acting as a regularization technique to prevent overfitting. The classification loss weight is retained at 0.8, while the box loss weight is 0.05, balancing the contributions of each component in the loss function.

However, for the training of the YOLOv11l model, a new patience parameter is introduced which is set to 20 epochs. This decision is influenced by the previous training behavior of the previous models, particularly the severe overfitting and NaN values encountered with YOLOv8l. By incorporating the patience parameter, the training process implements an early stopping mechanism, preventing unnecessary computations and avoiding performance degradation due to overfitting. If the validation loss does not improve for 20 consecutive epochs, the training process automatically halts, ensuring that only the best-performing model is retained.

For YOLOv11m, the training process was structured similarly to YOLOv8m segmentation. The model is trained for a maximum of 100 epochs while monitoring the loss curves and segmentation quality. Since YOLOv8m had shown some signs of overfitting but not to a severe extent, no explicit early stopping mechanism was used for YOLOv11m, given their comparable parameter sizes. However, to prevent unnecessary overfitting, checkpoints are taken at regular intervals to ensure that the best-performing model is preserved. The training loss curves for the YOLOv11m segmentation model exhibit a steady downward trend,

indicating a successful learning process. The segmentation loss shown in Figure 7.10a follows a downward trajectory, reflecting that the model is effectively learning to segment fractures from the background. However, towards the later epochs, there are some fluctuations, which might suggest the early signs of overfitting or sensitivity to small perturbations in the training data. The presence of slight spikes around the later epochs may indicate increased variance in learning, but the overall trend remains downward, which is a positive sign.

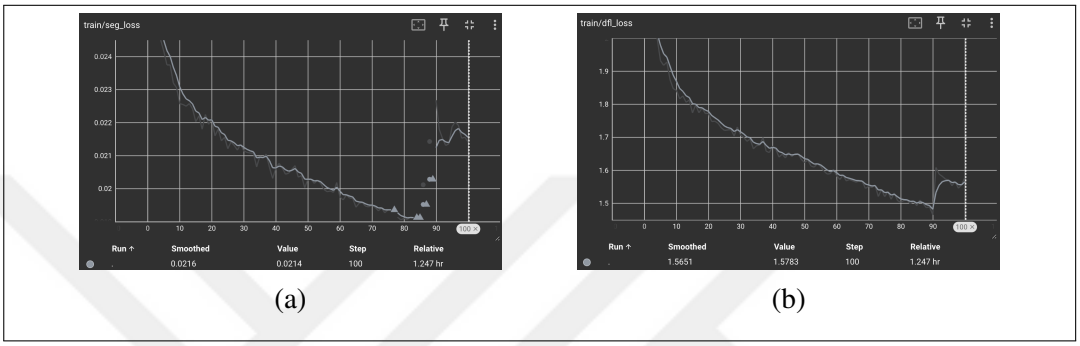
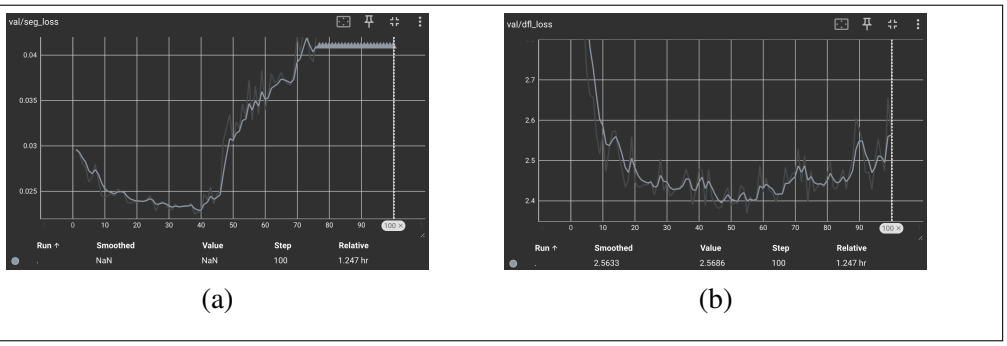


Figure 7.10. (a) YOLOv11m training segmentation loss, (b) training DFL loss.

The DFL loss, on the other hand, presented in Figure 7.10b, initially starts at a higher value and continuously decreases over the epochs, demonstrating that the model is progressively refining its object localization and bounding box predictions. This steady decline suggests that the model is improving in predicting the correct locations of fractures within the images. Thus, these results confirm that the model successfully learns from the dataset and minimizes both classification and segmentation errors.

The validation graphs for the model training are shown in Figure 7.11, providing a clear indication of overfitting as training progresses. The validation loss graphs for the YOLOv11m segmentation model provide a clear indication of overfitting as the training progresses. Initially, the validation segmentation loss follows a downward trend in Figure 7.11a, demonstrating that the model is learning to segment fractures effectively. However, after approximately 40 epochs, the validation segmentation loss begins to rise consistently, which is a strong sign that the model is overfitting to the training data.

Similarly, the validation DFL shown in Figure 7.11b presents a decreasing trend in the early stages of training, suggesting that the model is improving in learning the spatial distributions of fractures. However, after around 50 epochs, it begins to oscillate with increasing variance,



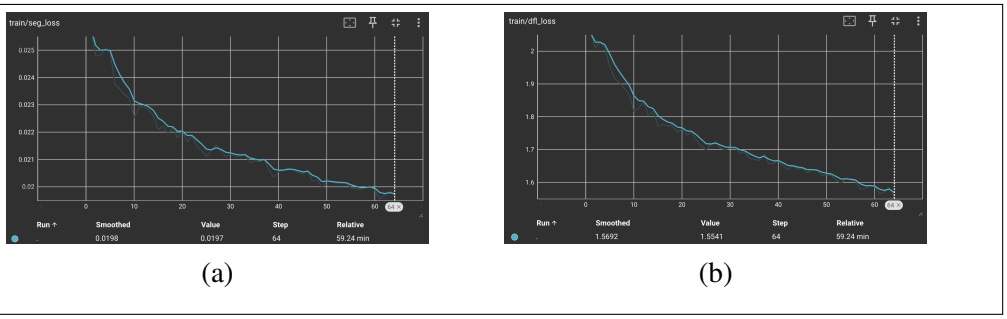
(a) (b)

Figure 7.11. (a) YOLOv11m validation segmentation loss, (b) validation DFL loss.

further confirming that the model is no longer generalizing well. This trend highlights the importance of checkpointing at the best-performing model to avoid deploying an overfit model.

Given these results, the checkpoint corresponding to the epoch where the validation loss was at its lowest is selected for testing, ensuring that the model used for evaluation has the best trade-off between learning and generalization. The observed overfitting pattern reinforces the need for tuning hyperparameters such as dropout, weight decay, or early stopping patience to achieve optimal results in future experiments.

The training loss graphs shown in Figure 7.12 for the YOLOv11l model illustrate a consistent decline in both the DFL and segmentation loss across epochs. The DFL loss, which measures the confidence in the predicted bounding box locations, starts at a high value and gradually decreases, indicating that the model is learning to refine its predictions. Similarly, the segmentation loss follows a downward trend, demonstrating that the model is progressively improving in delineating fracture regions.

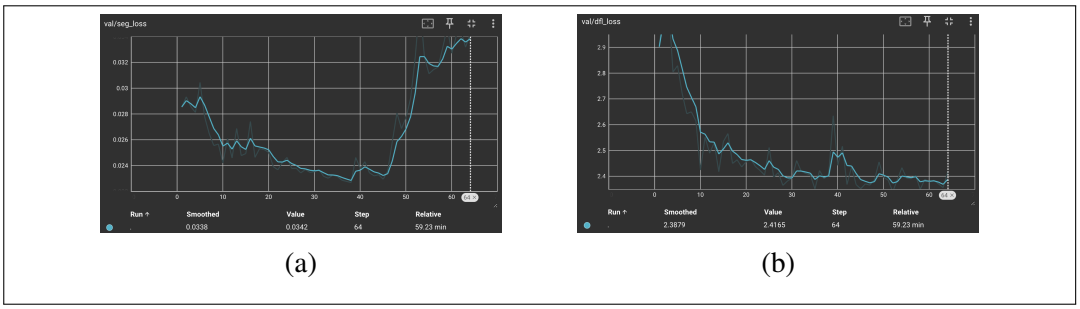


(a) (b)

Figure 7.12. (a) YOLOv11l training segmentation loss, (b) training DFL loss.

However, an important distinction in this training process is the implementation of an early stopping mechanism using a patience parameter of 20 epochs. The training loss graphs show that the model was stopped at 64 epochs, suggesting that further training did not yield significant improvement. This controlled termination of training optimizes generalization and prevents overfitting, which was a concern in previous models. Compared to other trained segmentation models, YOLOv11l exhibits a stable and steadily decreasing loss, aligning with the intended role of the patience parameter. This indicates that the model efficiently learned the dataset's features while avoiding excessive memorization of the training data. The best-performing checkpoint from this training will be used for the testing phase to ensure optimal segmentation accuracy.

The validation losses for the YOLOv11l segmentation model provide insight into how well the model generalizes to unseen data, as presented in Figure 7.13. The validation DFL follows an expected downward trend in the initial training phase, indicating that the model is effectively learning. However, after a certain point, the loss stabilizes and then fluctuates slightly. This behavior suggests that while the model improves initially, it reaches a plateau where additional training does not significantly enhance its performance. The validation segmentation loss follows a similar trajectory, initially decreasing but then sharply increasing in later epochs. This pattern is a clear indication of overfitting, where the model begins to memorize the training data rather than generalize well to new instances. The early stopping patience parameter, set to 20 epochs, plays a crucial role in mitigating overfitting. By allowing training to continue only if the validation performance continues to improve, the model avoids unnecessary over-training. In this case, training stopped automatically after 64 epochs.



(a) (b)

Figure 7.13. (a) YOLOv11l validation segmentation loss, (b) validation DFL loss.

The training of the YOLOv11m model took a total of 1.334 hours, whereas the YOLOv11l

model training took a total time of 1.08 hours due to the early halting of the training because of the patience parameter.

8. APPLICATION IMPLEMENTATION

This section presents the technical implementation of the Python-based application developed for deep learning-based medical image analysis. The primary aim of the application is to enable the inference of trained PyTorch models, either for object detection or semantic segmentation, on radiographic wrist images. Accordingly, the application incorporates an intuitive graphical interface that supports user interaction, feedback collection, and visual evaluation of predictions.

The prototype has been designed in a modular and model-agnostic manner, allowing integration with various types of architectures. Thus, the best-performing model can be imported into the application after various evaluation steps, whether it is detection or segmentation. Particularly, the application itself is structured into two main components, namely ”GUI” and ”predict”. The model and inference module is inside the predict component, which handles model loading, preprocessing, prediction logic, and overlay generation. It is designed to remain decoupled from the GUI logic, thus enabling reuse with different models. The algorithm for this component is presented in Algorithm 8.1.

Algorithm 8.1. Model inference workflow

Require: A trained PyTorch model checkpoint $\mathcal{M}$, and an input image path $\mathcal{I}$
1: Load the model $\mathcal{M}$ and set to evaluation mode
2: Load the image $\mathcal{I}$ and preprocess it (resize, normalize, convert to tensor)
3: Move preprocessed tensor to the appropriate device (CPU or GPU)
4: Perform inference: $P \Leftarrow \mathcal{M}$(tensor)
5: **if** model is a segmentation network **then**
6: Apply sigmoid activation to P and threshold to obtain binary mask
7: **else if** model is an object detector **then**
8: Decode output P into bounding boxes and labels
9: **end if**
10: Return the final prediction mask or detection result

The front end of the application is developed using PyQt6, with the goal of offering a responsive and user-friendly environment to medical professionals. The GUI is designed to streamline the workflow from image selection to model inference and result visualization. The sample design for the application is presented in Figure 8.1.

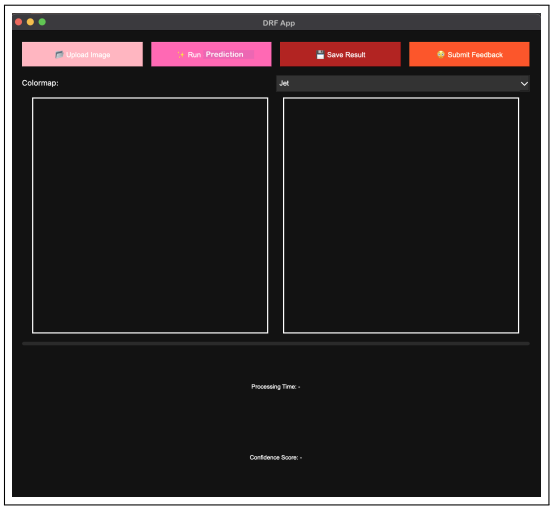


Figure 8.1. The Python application layout.

The image upload button lets the users select a radiographic image from their local filesystem. Supported formats include JPEG and PNG. The run prediction button, on the other hand, executes the selected model on the uploaded image, displaying the resulting segmentation mask or detection bounding boxes alongside the original image. The colormap selection is a pull-down menu that allows users to adjust the color scheme of the overlay (Jet, Bone, Plasma, etc.) for improved visualization and preference in the output image.

Furthermore, a progress bar is also implemented to provide the user a processing status feedback during the model inference. Lastly, at the bottom of the screen, the application displays the time taken to complete inference and the confidence score of the prediction, which allows users to evaluate model certainty and performance. Depending on the model used, the GUI supports both overlaying segmentation masks and drawing bounding boxes on the image. All visual outputs are rendered directly on the interface for immediate feedback.

Accordingly, the GUI component which implements the PyQt6 interface manages user interaction, threads model execution to avoid blocking the GUI, and supports post-inference visualization and feedback collection. Additionally, a background process thread is used to execute model inference, ensuring smooth UI responsiveness during heavy computation tasks. The GUI component workflow is presented in Algorithm 8.2.

The application loads a pre-trained model checkpoint saved in the PyTorch format. Upon initialization, the model is transferred to the available device, which could be GPU or CPU,

Algorithm 8.2. GUI workflow for user interaction and model inference

Require: A PyQt-based GUI interface, a pre-trained model, and an image selected by the user
Wait for the user to select an input image file
Display the selected image within the GUI layout
Allow the doctor to optionally submit pre-prediction feedback
Wait for the user to click the ‘‘Run Prediction’’ button
Start a background thread for model execution
In the background thread:
    Preprocess the selected image
    Perform model inference and obtain prediction
    Generate overlay visualization
    Display the overlay result within the GUI
Enable the ‘‘Edit Segmentation’’ and ‘‘Save Result’’ buttons
Allow the doctor to optionally submit post-prediction feedback

and set to evaluation mode. The model's class structure and preprocessing requirements are encapsulated in a dedicated Python module to ensure maintainability and flexibility.

After testing and evaluation, the model is to be selected. Thus, based on the selected model's requirements, the preprocessing pipeline is constructed to adapt dynamically. However, generic preprocessing operations include input resizing, normalization based on the architecture's training configuration, tensor conversion, and appropriate device transfer. This flexibility allows the system to support any model trained with consistent preprocessing procedures, regardless of whether it performs binary segmentation or detection. Following preprocessing, the input image is passed to the model for prediction. Postprocessing steps, such as applying Sigmoid activations, softmax, confidence thresholds, or converting detection outputs into bounding boxes or masks are also handled according to the model type.

Another core component of the application is the integration of a user feedback system, which facilitates human-in-the-loop correction and the potential for future model fine-tuning. Before or after the inference, the users can use the ”Leave Feedback” button to leave a comment about the application in general or report case of a false prediction. The feedback gets encrypted along with the metadata and saved into a log file. Only admins can decrypt the log data with the Fernet key, allowing integration of medical users' knowledge for future training pipelines. However, it should also be highlighted that radiographs uploaded into the application are

not saved at any point, to prevent any privacy-related issues in such a sensitive domain. This feedback approach is only designed to collect the problems related to false predictions from the experts so that they can be used to improve model performance through re-training, addition of new data, or fine-tuning.

Last but not least, to facilitate usability and portability, the application is configured for deployment as a standalone executable. Using packaging tools such as Python installer, the entire application, including model checkpoints and required libraries is bundled and distributed to end users without requiring them to manually configure a Python environment.

9. EVALUATION AND TESTING OF THE DETECTION MODELS

The flowchart illustrated in Figure 9.1 shows the undertaken steps to evaluate and test trained object detection models. The process is divided into three main phases —namely (i) introduction, (ii) model-specific analysis, and (iii) comparison. The first phase involves describing the methodology applied for the assessment of detection models and establishing conceptual clarity before proceeding to more specific model sub-evaluations.

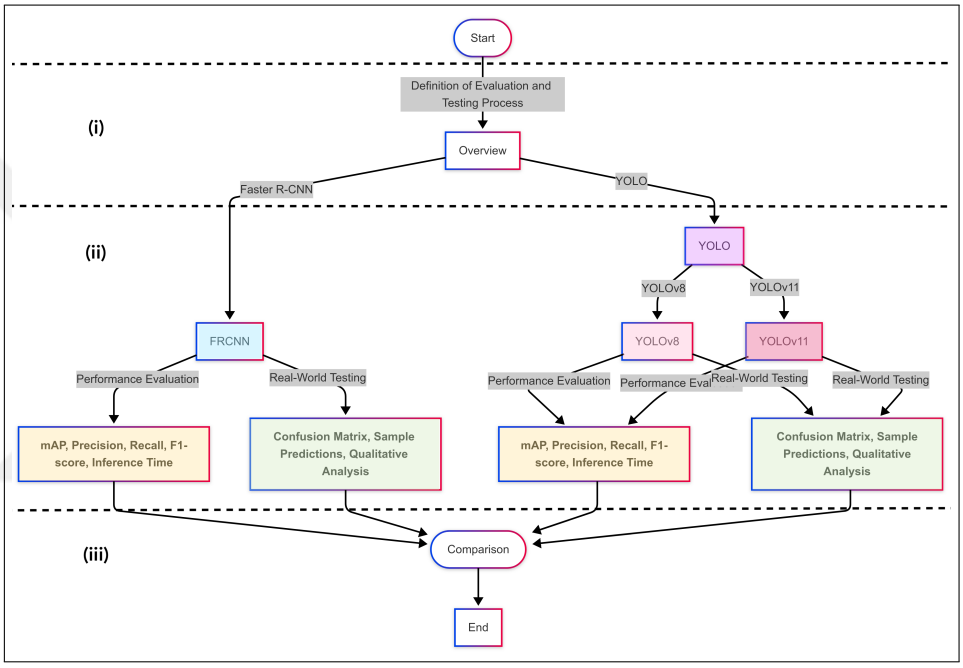


Figure 9.1. Flowchart of the procedure for evaluating and testing the detection models.

In model-specific assessment, Faster R-CNN and YOLO-based models are individually evaluated and tested. Each model undergoes through two key phases; performance evaluation and real-world testing. The performance evaluation procedure utilizes quantitive metrics such as mAP, precision, recall, F1-score, and inference times to analyze the models. Real-world testing, on the other hand, involves testing the models on a separate dataset to assess their ability to generalize. This phase is evaluated using confusion matrices, sample predictions, and quantitive analysis to examine how well the models identify fractures in new, unseen radiographs. Finally, the comparison part consists of synthesizing the evaluation and testing results to compare models and identify the best-performing detection approach based on various metrics.

9.1. FASTER R-CNN PERFORMANCE EVALUATION

Faster R-CNN is a widely recognized object detection model that employs a region-based approach that offers a balance between detection accuracy and computational efficiency. This section provides a comprehensive evaluation of the model's performance using both training metrics and validation results to assess its effectiveness in detecting DRFs.

The validation set is utilized to compute detection metrics, including bounding box average precision at different thresholds, recall, and F1 score to examine the detection performance of the model appropriately. As mentioned previously in Chapter 6, the training is conducted for 8,000 iterations with validation metrics computed at every 100th iteration, resulting in a total of 80 evaluation steps. This periodic validation facilitates the tracking of performance trends throughout training, enabling the identification of improvements, stability, or potential degradation in detection accuracy. The subsequent analysis aims to provide a detailed understanding of Faster R-CNN's detection capabilities based on numerical information, highlighting its strengths and limitations within this specific application.

The bounding box average precision shown in Figure 9.2 represents the model's detection accuracy over the 80 evaluation steps that span 8,000 training iterations at various IoU thresholds. At the beginning of the training, for approximately up to 1,500 iterations, the AP value remains close to zero, indicating that the model was unable to correctly detect objects. However, a gradual increase in AP is observed, with a significant improvement occurring between 1,500 and 5,000 iterations, which suggests that the model is effectively learning object features and refining its bounding box predictions.

After around 5,000 iterations, the AP curve begins to flatten, indicating that the model has reached a near-optimal detection performance. Furthermore, the AP stabilizes around 15%, which indicates a ceiling effect where further training does not significantly enhance performance. While small oscillations in AP are visible after stabilization, they may be due to the complexity of certain samples in the validation set. However, the final AP of 15% is relatively low, which suggests that despite tedious hyperparameter tuning and a large number of iterations, the model may struggle to detect fractures from the radiographs later on.

The bounding box average precision is also evaluated at specific IoU thresholds to determine

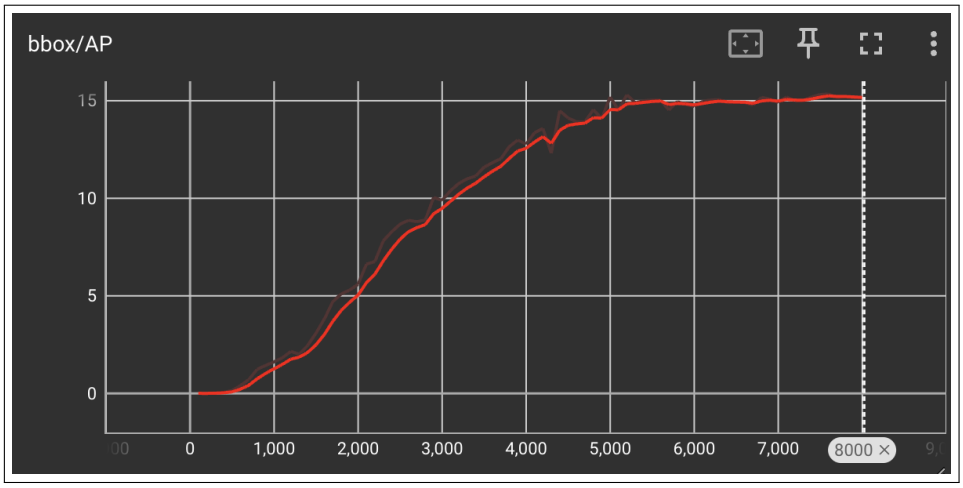


Figure 9.2. Faster R-CNN bounding box precision at different IoU thresholds.

how well the model detects and localizes objects under varying levels of stringency. Figure 9.3 shows the AP when IoU is larger or equal to 0.5 such that if the ground truth and the prediction overlap at least by 50%, the prediction is considered to be true. Throughout the training, the AP50 curve steadily increases and reaches approximately 45% AP at 8,000 iterations, which indicates it performs well when a relatively loose IoU threshold is used.

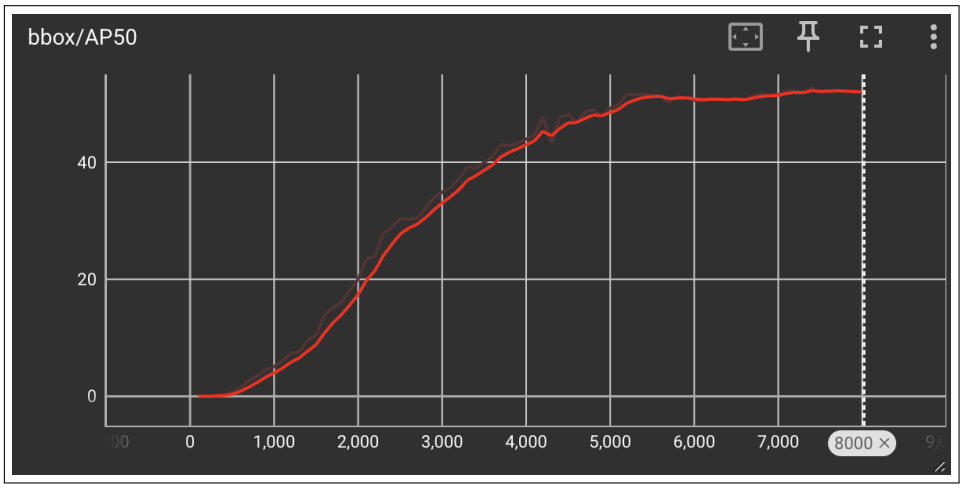


Figure 9.3. Faster R-CNN bounding box precision at IoU ≥ 0.5.

Figure 9.4, on the other hand, presents the AP where the IoU threshold is set to be larger or equal to 0.75. AP75 starts very low and gradually improves over time, reaching just above 3% at the end of training. Moreover, the fluctuations in the graph indicate that the model struggles with achieving high-precision localization if the IoU threshold is high. Unlike AP50, AP75 does not plateau as smoothly. The detection performance of AP at IoU 50

metric is significantly higher compared to the stricter AP at 75 metric, indicating the model is good at general fracture detection but may struggle with precise localization.

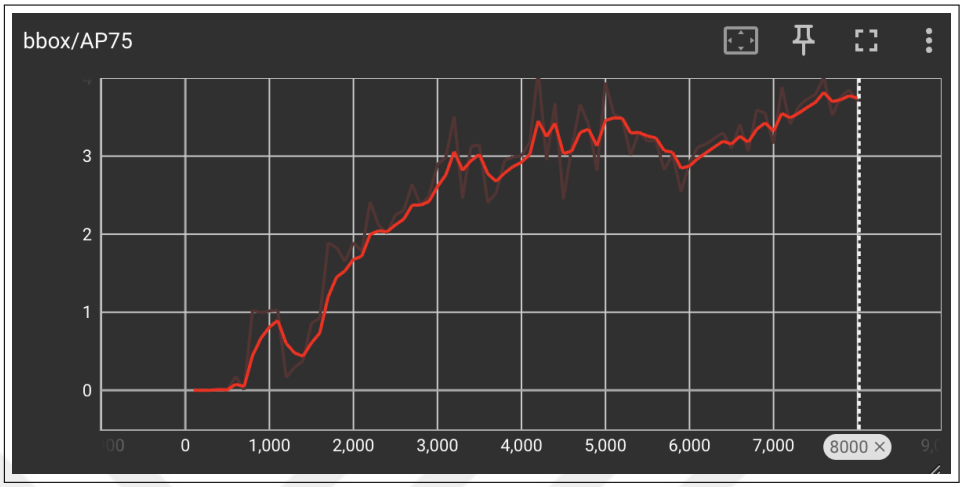


Figure 9.4. Faster R-CNN bounding box precision at IoU ≥ 0.75.

While the AP at IoU 50 is approximately 45% and indicates that the model is reasonably effective at detecting fractures, AP at 75 is significantly lower at approximately 3%, which suggests that the model struggles with the localization of fractures. The significant gap between these two metrics might suggest that predicted bounding boxes are not refined adequately and the dataset at hand is not enough for Faster R-CNN model training to achieve accurate localization.

The impact of fracture sized are compared and accordingly, Figure 9.5a shows AP for small ground truth boxes (APs), Figure 9.5b presents AP for medium ground truths (APm), and Figure 9.5c displays the AP for larger ground truths (APl). Detecting subtle fractures is a well-known challenge, as smaller or less distinct instances are significantly more difficult to identify compared to larger, more prominent ones. This analysis is crucial in identifying whether the Faster R-CNN model has difficulty detecting smaller, more subtle fractures.

The APs curve starts at zero and progressively increases, reaching approximately 6% AP at 8,000 iterations. While the curve exhibits some fluctuations around 4,000 to 5,000 iterations, it slowly stabilizes towards the end of the training. While the regular improvements in AP indicate that the model has learned some patterns, the final APs metric is relatively low.

The APm metric for medium size fractures is relatively higher compared to APs, which is for

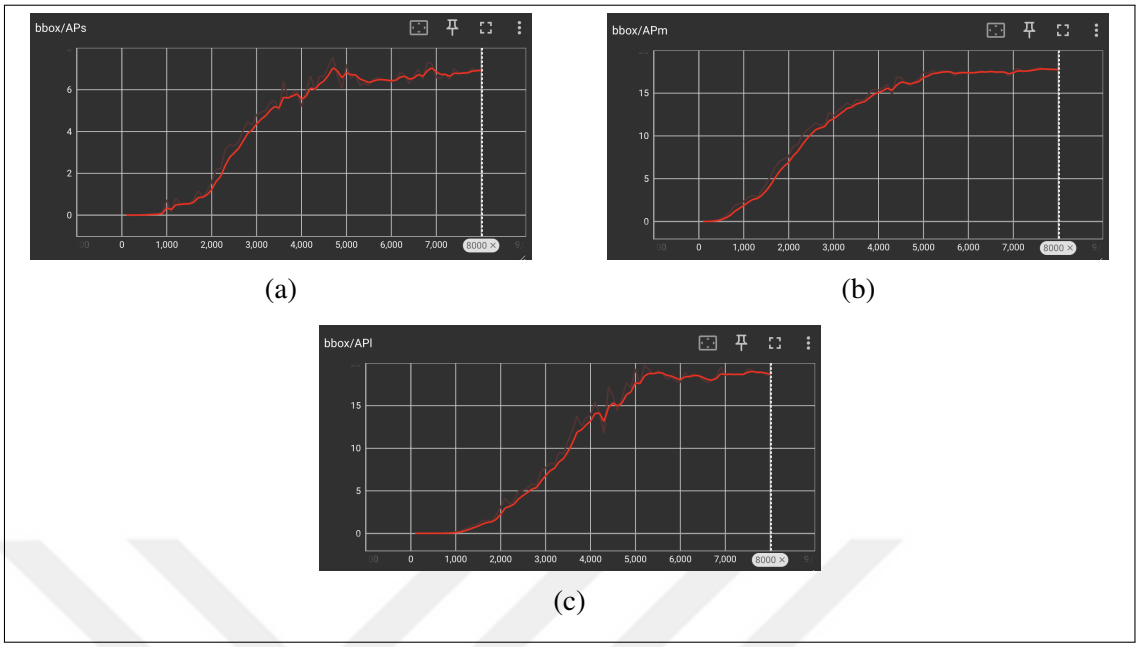


Figure 9.5. (a) Faster R-CNN AP on small, (b) medium, and, (c) large fractures.

small fractures, following a similar trend to the overall APs curve and reaching 17% at the final iteration. Performance increases steadily, with a sharp rise up to 5,000 iterations, after which improvements slow down and the model shows a much better detection capability for medium-sized fractures compared to small fractures. While the model performs better on medium-sized fractures, its detection accuracy is still not optimal. Furthermore, some fractures may appear larger due to swelling, contrast differences, or projection angles, leading to higher AP scores in APm. However, the fact that APm is significantly higher than APs suggests that the model is more reliable for larger or more prominent fractures.

Finally, the APl curve in Figure 9.5c yields the highest AP value, reaching approximately 19% by the end of the training. Moreover, the learning progression is smooth and consistent, which suggests that the model can detect large fractures more efficiently. While there are minor fluctuations in the later stages, the overall trend is not significantly impacted. This means that the model performs best on large fractures, which is expected since larger fractures provide more visual cues for detection. However, it is the fact that DRFs are mostly smaller because of their nature, which makes this AP less relevant to the primary task. This further reinforces the fact that the Faster R-CNN option is not suitable for this task with the dataset at hand, as the majority of clinically relevant findings in DRF radiographs fall into the small object category.

Figure 9.6 shows a combination of all key evaluation metrics to evaluate the Faster R-CNN object detection model. The AP at IoU 0.50:0.95, denoted with a yellow line, measures the mean average precision across a range of IoU thresholds, namely 0.5 to 0.95 with a step size of 0.05. It provides a comprehensive measure of how well the model detects objects while ensuring precise localization and is a stricter metric compared to AP50, as it accounts for varying levels of IoU stringency. The AP starts at 0 where the IoU threshold is equal to 0.95 and increases steadily over time, stabilizing at 0.15. The AP at 0.5 was discussed previously, denoted with an orange line.

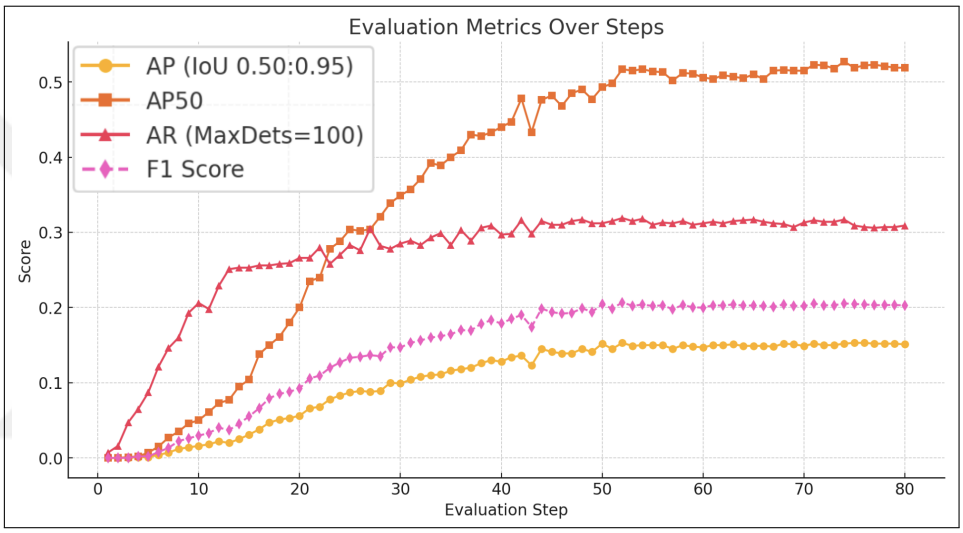


Figure 9.6. Key evaluation metrics graph for Faster R-CNN evaluation.

The average recall (AR), denoted with a red line, represents the proportion of true objects that the model successfully detects with a maximum of 100 detections per image. The AR metric rises quickly in the early stages and plateaus at approximately 30%. There are not many fluctuations, which indicates that there is no significant instability in how well the model consistently detects fractures. However, the recall value remains relatively low at approximately 32%, indicating that the model is missing a substantial number of fractures and producing a high number of false negatives. The stability of the curve indicated that the model was successful in learning a pattern, however, can not detect all of the fractures, probably missing the smaller ones.

Finally, the F1-Score, shown with a pink line, exhibits a similar steady improvement, reaching around 15% in the end. The curve being smoother than the AR curve suggests a balanced trade-off between precision and recall, confirming that they are both improving together, yet,

the overall performance remains low. Since recall is low, the F1-score also remains low, reinforcing that the model might struggle to detect all fractures.

Lastly, the graph in Figure 9.7 presents the inference speed per iteration over 80 validation steps, measured in seconds per iteration (s/iter). Understanding inference time is crucial for assessing whether the Faster R-CNN model is computationally viable and suitable for real-time or near-real-time clinical solutions. The average inference speed is stable around 0.036 s/iter during the initial validation steps, which suggests that the model maintains a consistent computational load without major slowdowns. However, the sudden spikes in later validation steps can be attributed to variations in input image complexity, irregular GPU load balancing, memory bottlenecks or longer bounding box proposals for certain images, which lead to increased processing time. Furthermore, the occasional sips in inference time suggest that some validation steps require less computation, indicating that certain images might have had fewer detected objects or required fewer region proposals, resulting in faster predictions.

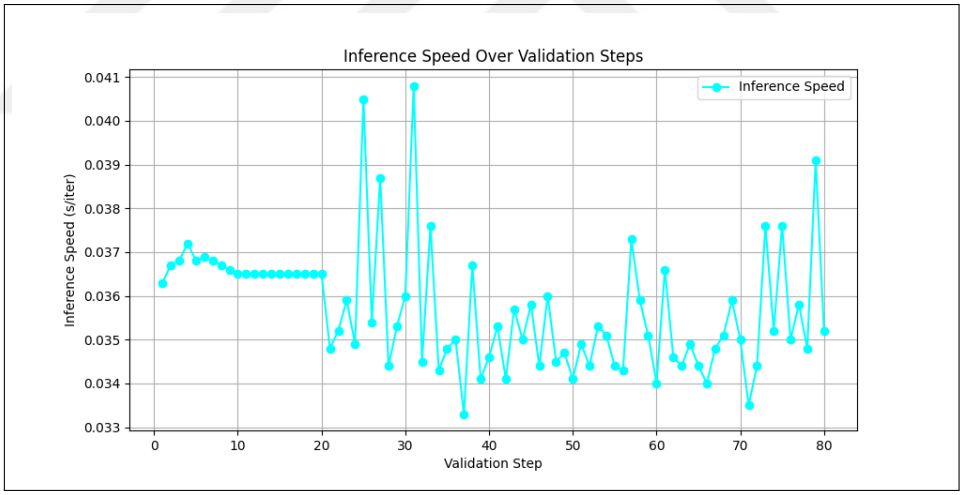


Figure 9.7. Faster R-CNN inference speed per iteration on the validation set.

In conclusion, Faster R-CNN maintains stable inference times, making it viable for batch-processing applications and the inference spikes suggest computational inefficiencies, which could slow down predictions for complex cases. However, the speed of the model does not compensate for the lack of precision or accuracy, as it still may not be ideal for high-speed diagnostics.

9.2. FASTER R-CNN REAL-WORLD TESTING

To assess the real-world applicability of the Faster R-CNN model, an external test set of 100 radiographs is utilized, containing a total of 110 fracture instances. Unlike the training and validation phases, where model performance is monitored using standard evaluation metrics, this test set evaluates the model's ability to detect fractures in a clinical scenario. Each radiograph contains at least one fracture, and in some cases, multiple fractures are present in one radiograph.

The model's predictions are analyzed by comparing them against ground truth annotations obtained from an orthopedic surgeon, and a confusion matrix is constructed to visualize the detection outcomes. The confusion matrix categorizes model predictions into true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) to provide a detailed breakdown of detection accuracy, as presented in Figure 9.8 where all predictions are produced with an IoU threshold of 0.5.

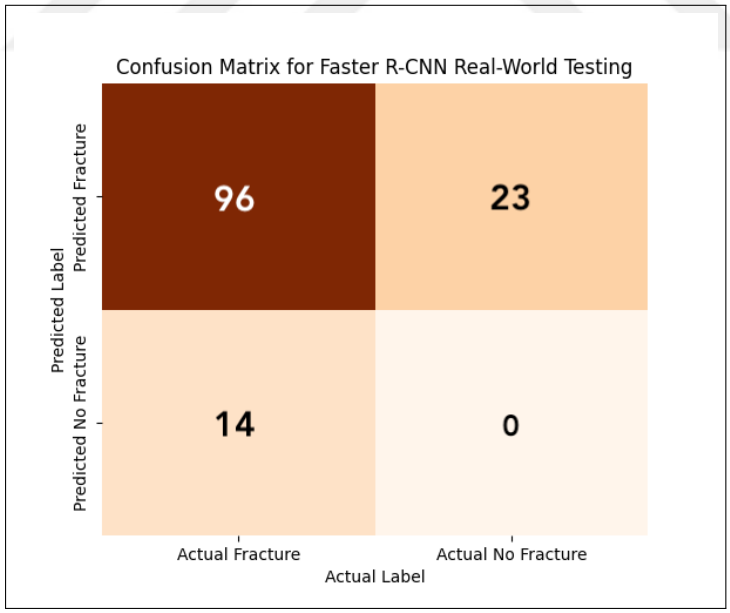


Figure 9.8. Faster R-CNN confusion matrix on the test set.

The true positive number is 96 which means that the model correctly identified 96 fractures. This indicates that the model is able to detect a large portion of actual fractures. However, the false positive count is relatively high with 23 in total, meaning that the model predicted 23 extra bounding boxes that did not correspond to actual fractures. This suggests that the model

may be over-detecting fractures in some cases, which might lead to false alarms. Lastly, the false negative is valued at 14 and these instances are the actual fractures that are missed by the model. This is a critical issue in medical imaging since missed fractures could lead to misdiagnosis or delayed treatment. True negatives are not applicable in this case, as every radiograph in the test set contains at least one fracture.

To quantitatively assess the Faster R-CNN model's performance, three key metrics, namely precision, recall, accuracy, and F1-score are computed based on the confusion matrix. Table 9.1 presents the computed evaluation metrics from the confusion matrix as values and as percentages.

Table 9.1. Faster R-CNN performance metrics on test set

Metric	**Formula**	**Value**	**Percentage**
Precision	TP / (TP + FP)	0.8076	80.76%
Recall	TP / (TP + FN)	0.8727	87.27%
F1-Score	2 * (P * R) / (P + R)	0.8387	83.87%
Accuracy	TP / (TP + FP + FN)	0.7218	72.18%

As precision is the positive predictive value and measures how many of the predicted fractures are actually correct, a precision score of 80.76% indicates that while the model is fairly reliable, it has a tendency to identify non-existent fractures, leading to false positives. Recall, on the other hand, calculates how many actual fractures the model successfully identified. While the model is somewhat sensitive, it failed to detect 12.73% of the actual fractures overall. F1-Score balancing the precision and recall suggests that the model performs alright overall, however, the missed detections remain a crucial concern.

While accuracy is not the most meaningful metric due to the lack of explicit true negatives, it is still computed for comparison reasons with the other models. Accordingly, Table 9.1 shows the simplified formula of the standard accuracy where TN is removed. 72.18% accuracy means that, out of all fracture-related predictions, the model made a correct detection 72.18% of the time.

Additional in-depth analysis of false positives and false negatives is conducted to reveal notable challenges the model struggled with. The potential false positive challenges are grouped into three distinct categories, namely bounding box overlap and mislocalization,

normal bone structures that resemble fractures, and with no apparent cause.

The most common source of false positives is the mislocalization of actual fractures, where the model generated multiple overlapping bounding boxes around a single fracture. 16 cases of false positives are marked under this category. In such cases, the model correctly detected the fracture but assigned an additional bounding box nearby with a lower confidence score. This is a common issue in region-based models like Faster R-CNN, where anchor boxes may generate multiple overlapping proposals for the same object. A more aggressive IoU threshold for merging overlapping predictions could mitigate this issue. However, after experimenting with lower IoU thresholds, it was observed that the model struggles to make as many correct predictions as it does with the current IoU threshold. Thus, this approach was not preferred. Figure 9.9 presents a sample case where an extra bounding box is generated within the correct bounding box.



Figure 9.9. False positive due to overlapping bounding boxes.

The second most common cause of false positives stems from the model mistakenly identifying finder bones as fractures in AP view X-rays specifically. 4 cases of false positives show that the model detects the sharp cortical edges as joint spaces as potential fracture sites, indicating that the model is too sensitive to normal anatomical variations, leading to incorrect fracture predictions consequently. In Figure 9.10, such a case is shown, where the model

makes an additional detection on the hand.



Figure 9.10. False positive due to high sensitivity to normal bone structure.

For the remaining 3 cases, the false positives had no clear visual justification. This could be attributed to suboptimal feature learning where the model assigns a fracture label to patterns it does not fully recognize or random noise in the image influencing the model's decision.

A total of 14 fractures are missed, highlighting important limitations in the model's ability to capture subtle or complex fracture patterns. These false negatives were categorized into 4 different categories, namely casts, low contrast images with casts, surgical hardware, and subtle fractures. However, the most notable cases are missed due to casts, 5 cases to be exact. This suggests that the model struggles to distinguish fractures when overlying materials obscure the fracture site. Casted radiographs create additional noise in the image due to varying radiopacity, and reduced visibility of fracture lines, leading to an increased likelihood of being ignored by the detection model. Nevertheless, the dataset consisted of enough number of casted radiographs, which suggests that the additional noise impacts the model's decisions. Furthermore, 2 fractures were further missed due to a combination of casting and low contrast. This indicates that the model relies heavily on clear fracture visibility, and low-contrast images reduce its ability to differentiate fractures from normal bone. Figure 9.11 exhibits a sample missed fracture by the model where a cast is involved and the contrast is low.



Figure 9.11. False negative due to casts and low contrast.

The other most common cause of missed fractures involve small or faintly visible fracture lines in 5 cases. Here, the key weakness of the Faster R-CNN model is highlighted. The model likely fails to capture extremely fine details, leading to missed detections for subtle fractures. This aligns with earlier findings where AP for small objects is noticeably low, confirming that subtle fractures are particularly challenging. Despite the utilization of feature pyramid networks, small object detection remains a challenge for the Faster R-CNN model.

A total of 2 missed cases are due to the presence of orthopedic hardware such as screws or plates. This implies that the model may be confusing metallic implants with actual bone structures. However, this issue is particularly relevant for post-surgical fracture assessments, where hardware is frequently present. Figure 9.12 shows an example of missed prediction by the model where surgical hardware is present.



Figure 9.12. False negative due to surgical hardware.

9.3. YOLO DETECTION PERFORMANCE EVALUATION

Given the limitations observed in Faster R-CNN, particularly in detecting small fractures and handling complex radiographs with casts or implants, this section aims to evaluate and determine whether YOLO-based architectures offer better generalization and performance for the detection task. By comparing YOLOv8 and YOLOv11 models in both medium and large configurations, the review will provide insights into the trade-off between detection accuracy and inference speed across different YOLO versions, the impact of model size (m vs. l) on fracture detection performance, and whether YOLOv11 models offer improvements over YOLOv8.

The following sections present a detailed performance evaluation of each YOLO model, highlighting its strengths, limitations, and comparative effectiveness in fracture detection for real-world clinical applications.

9.3.1. YOLOv8 Detection Evaluation

The YOLOv8m model, being a medium-scale YOLO model, is designed to balance detection accuracy and computational efficiency. To evaluate its detection performance on the fracture dataset, three key graphs are analyzed first, calculated on the validation set, namely the recall-confidence curve, precision-recall curve, and the precision-confidence curve. These graphs provide insights into how well the model performs across different confidence thresholds, its trade-offs between precision and recall, and its overall detection quality.

The Recall-Confidence curve illustrated in Figure 9.13 shows the relationship between recall and the confidence threshold used for detection. A high recall indicates that the model successfully detects most fractures, while a decline in recall at higher confidence thresholds suggests that stricter confidence settings reduce the number of detections. The highest recall value of approximately 69% is achieved at very low confidence thresholds, meaning the model is detecting nearly all fractures at this stage. As the confidence threshold increases recall steadily declines.

When confidence is set above 0.8, recall drops to nearly zero, indicating that very few

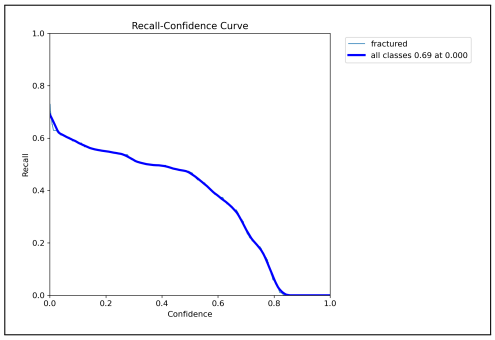


Figure 9.13. YOLOv8m model recall-confidence graph on the validation set.

detections are being made at strict confidence levels. This indicates that the model is highly sensitive to fractures at low confidence thresholds. However, as confidence increases, many fractures are missed, meaning the model tends to be overly cautious at higher confidence values. At higher confidence thresholds, the model is likely to suffer from low recall, leading to an increased false negative rate.

The precision-recall curve shown in Figure 9.14 is a critical measure of the trade-off between detection accuracy and sensitivity. At high recall values such as 0.8, the precision remains low, approximately 0.3, indicating that when the model attempts to detect as many fractures as possible, it produces a large number of false positives. The best precision-recall balance is achieved when the recall is valued at 0.5 and the precision remains relatively stable. Likewise, at low recall values, the precision is very high, which indicates that when the model is selective, it is very confident about its detections.

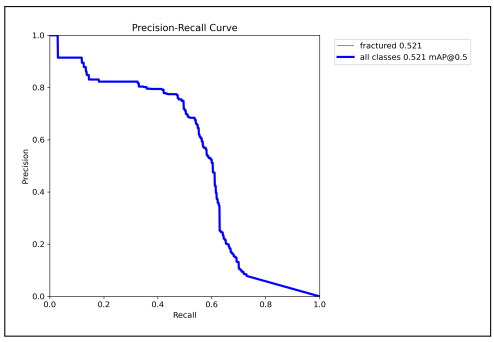


Figure 9.14. YOLOv8m model precision-recall curve on the validation set.

Thus, the model performs well when prioritizing high-confidence detections, with strong precision at lower recall values. However, to maximize recall, the model sacrifices precision, producing more false positive detections. For clinical applications, this suggests that

confidence thresholding will need to be optimized carefully to balance precision and recall.

Lastly, the precision-confidence curve presented in Figure 9.15 presents how precision varies across different confidence thresholds. Precision steadily increases as confidence rises, reaching its peak at about 0.84 confidence threshold, where precision is nearly 100%. However, at lower confidence values, precision drops significantly, indicating that many incorrect detections are made when confidence filtering is not applied. This suggests that applying a confidence threshold of approximately 0.8 ensures accurate detections, nonetheless, at this threshold, the recall would be significantly lower, leading to missed fractures. Thus, a balance is required to be found where precision is maximized without excessively sacrificing recall.

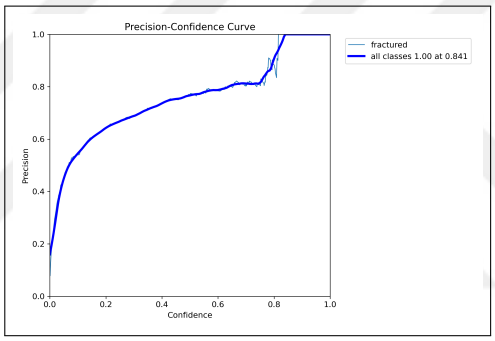


Figure 9.15. YOLOv8m model precision-confidence graph on the validation set.

To further assess the detection performance of the YOLOv8m model, the confusion matrix and F1-score curve provide additional insights into how well the model balances correct detections, false positives, and false negatives. The F1-score curve evaluates the balance between precision and recall across different confidence thresholds. The model's highest F1-score is 0.6 at a confidence threshold of 0.254, as presented in Figure 9.16. The F1-score increases rapidly as confidence rises from 0 to 0.25, peaking at 0.6. After 0.25 confidence, the F1-score plateaus slightly before declining beyond 0.6 confidence. At very high confidence thresholds, the F1-score drops to nearly zero, meaning the model makes almost no detections.

This indicates that the optimal confidence threshold for a balanced model is approximately 0.25 where precision and recall are both maximized together. At higher confidence thresholds, the model might become overly selective, reducing recall significantly, and leading to a situation where most fractures go undetected. However, if the confidence

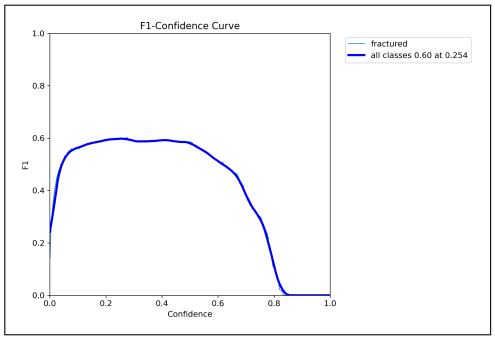


Figure 9.16. YOLOv8m F1-Score across confidence thresholds on the validation set.

threshold is set smaller than necessary, the model might produce more false positives. The model will be examined at this confidence threshold, as well as a higher confidence value at the testing section.

In Figure 9.17, the normalized confusion matrix is presented, showing the proportion of correct and incorrect classifications made by the YOLOv8m model in a scaled format, where each value represents a percentage rather than an absolute count. Based on the confusion matrix, 85.5% of the predicted fractures are correctly classified, while 14% of predicted fractures are actually background, corresponding to false positives. On the other hand, 100% of the predicted background cases were incorrectly classified as fractures, which is expected since the dataset contains only fractured radiographs, meaning there were no true background cases for the model to learn from. This suggests that while the model is able to correctly identify most fractures, it struggles with precision, as a notable proportion of predicted fractures are actually false positives. The high false positive rate indicates that the model tends to over-predict fractures, which could be attributed to sensitivity tuning. Compared to the Faster R-CNN model previously evaluated, YOLOv8m demonstrates improvements in fracture detection in some aspects but still presents significant challenges in balancing sensitivity and specificity.

The provided graphs in Figure 9.18 represents the training and validation performance of the YOLOv8m detection model over 100 epochs. The first row consists of training losses and key performance metrics, while the second row displays validation losses and evaluation metrics. In the first row, the training losses demonstrate a steady decline over the training period, indicating that the model is learning effectively and refining its bounding

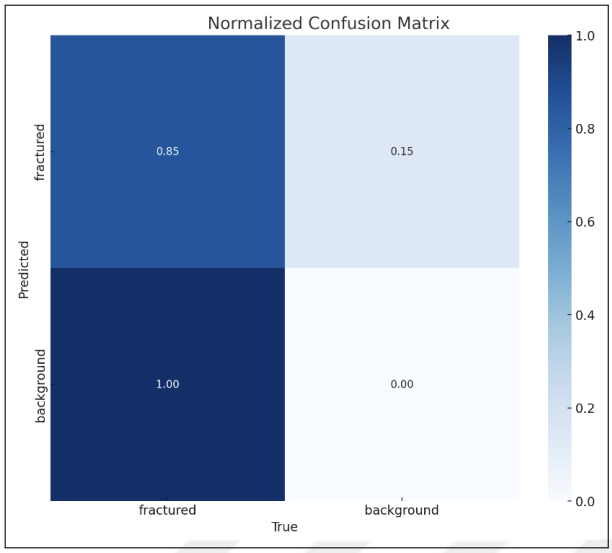


Figure 9.17. YOLOv8m model confusion matrix on the validation set.

box predictions. The box loss, which represents the model's ability to fit bounding boxes around fractures, decreases smoothly, suggesting that the model is improving its localization capability. Similarly, the classification loss and distribution focal loss decrease consistently, indicating that the model is becoming more confident in distinguishing between fractured and non-fractured regions.

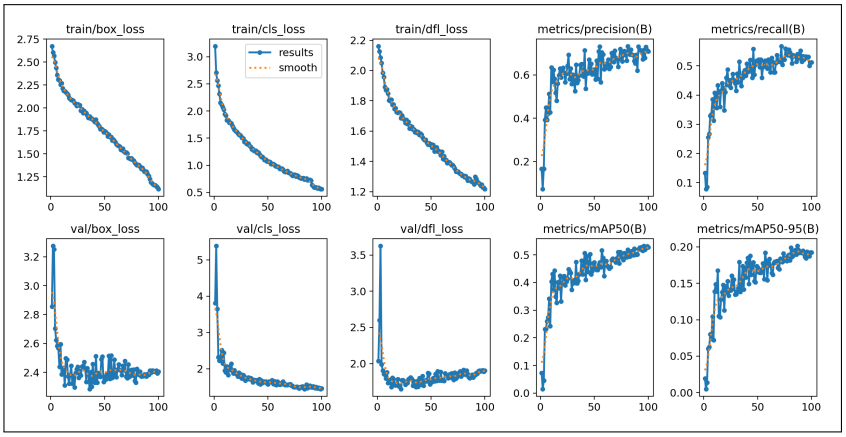


Figure 9.18. YOLOv8m results during training and on the validation set.

In the second row, the validation losses initially exhibit high values but rapidly stabilize after the first few epochs. The losses for validation are higher than their training counterparts, indicating some degree of overfitting, but they remain relatively stable after the initial fluctuations. The mAP50 and mAP50-95 both show a positive trend, confirming that the model is progressively improving its detection accuracy over training. However, the

mAP50-95 remains relatively low, which suggests that the model struggles to precisely localize fractures across different IoU thresholds.

The precision and recall metrics in the first row show a general upward trend, with recall experiencing more fluctuations than precision. This suggests that while the model becomes increasingly confident in its positive predictions, it still struggles to consistently detect all fractures, which aligns with the high false negative rate observed in the confusion matrix. The recall metric, while improving, does not reach a high value, reinforcing the observation that the model misses a significant number of fractures.

In conclusion, the best-performing YOLOv8m model, extracted at the optimal checkpoint during training, achieved the detection metrics shown in Table 9.2.

Table 9.2. Validation metrics for YOLOv8m

Metric	**YOLOv8m**	**YOLOv8m (%)**
Precision	0.662	66.2%
Recall	0.541	54.1%
mAP@50	0.521	52.1%
mAP@50-95	0.201	20.1%
F1-Score	0.595	59.5%

In Figure 9.19a the precision-confidence curve for the YOLOv8l model is shown. Precision-confidence curve shows that initially, at very low confidence thresholds, the model has a low precision, meaning a large number of false positives are included in the predictions. However, as confidence increases, precision steadily rises, reaching its peak at the highest confidence levels. This trend indicates that the model is more reliable when making high-confidence predictions, but at lower confidence values, it tends to produce more incorrect classifications. The optimal balance appears around a confidence threshold of 0.85, where the model reaches its best trade-off between making confident predictions and maintaining a high precision rate.

In Figure 9.19b, the precision-recall curve further reveals the trade-off between these two metrics. At high recall values, where the model captures most fractures, precision starts to drop significantly. This is expected, as increasing recall typically results in a greater number of false positives. The curve demonstrates a gradual decline in precision as recall

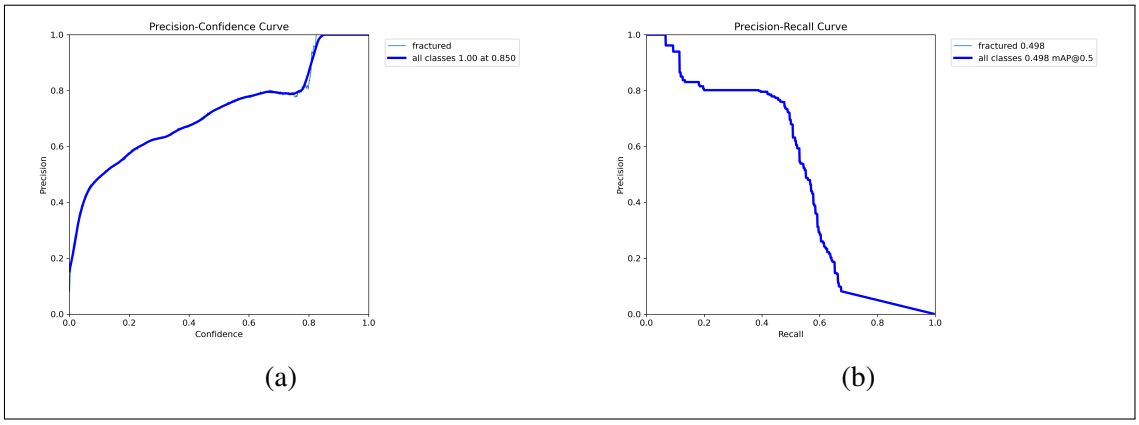


Figure 9.19. (a) YOLOv8l precision-confidence, (b) precision-recall curves on validation set.

increases, confirming that the model struggles to maintain both high recall and high precision simultaneously. The mean mAP@0.5 is approximately 0.498, which indicates that while the model achieves moderate precision across different recall thresholds, it does not maintain exceptionally high accuracy when aiming for complete fracture detection.

The F1-Confidence Curve shown in Figure 9.20a helps determine the best confidence threshold by balancing precision and recall. The F1-Score reaches its peak at a confidence threshold of 0.48, suggesting that setting the model's confidence threshold at this level would yield the best overall balance between detecting fractures and avoiding false positives. At confidence levels higher than this, precision improves, but recall begins to drop rapidly, reducing the model's ability to detect fractures comprehensively. Conversely, at lower confidence thresholds, recall is higher, but the increase in false positives reduces the F1-Score.

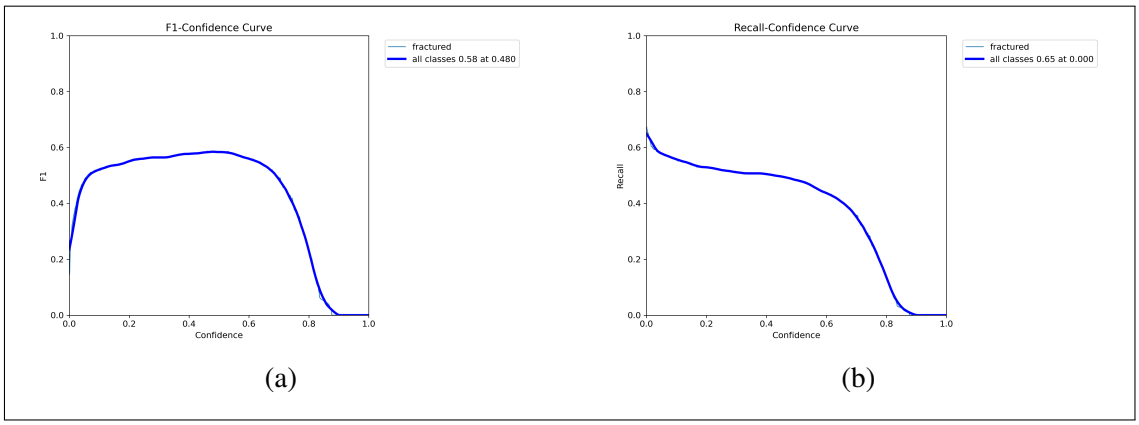


Figure 9.20. (a) YOLOv8l F1-Score-confidence, (b) recall-confidence curves on validation set.

Lastly, the recall-confidence curve presented in Figure 9.20b highlights how recall declines as the confidence threshold increases. At very low confidence levels, recall is relatively high, meaning the model detects a large proportion of fractures. However, as confidence increases, recall declines steadily, reaching near zero at the highest confidence levels. This drop-off indicates that setting an excessively high confidence threshold would lead to many missed fractures, as the model would become overly conservative in making predictions.

In Figure 9.21, the normalized confusion matrix for the YOLOv8l detection model on the validation set is presented. The YOLOv8l model correctly identifies fractures with a precision of 85%, meaning that when the model predicts a fracture, it is correct most of the time. However, its recall is significantly lower at 56%, indicating that it fails to detect a notable portion of actual fractures, leading to a substantial number of false negatives. This suggests that while the model is good at avoiding false alarms, it sacrifices sensitivity, which can be problematic in a clinical setting where missing a fracture is far more critical than incorrectly flagging a normal radiograph. Thus, the F1-score is derived to be 68% based on the values provided by the confusion matrix. Accuracy, on the other hand, is 51%.

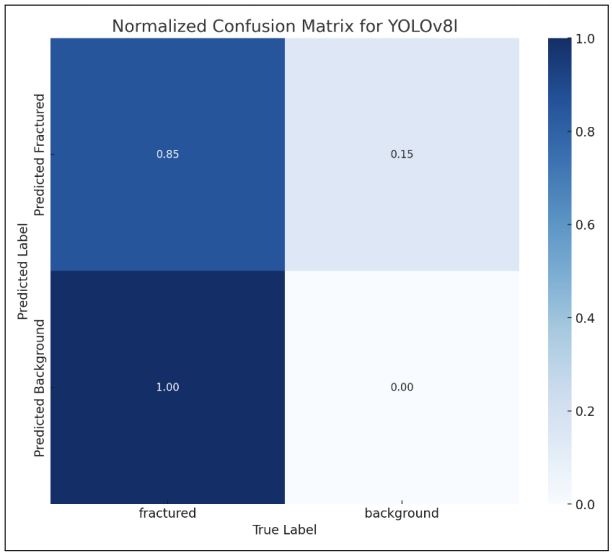


Figure 9.21. YOLOv8l normalized confusion matrix.

The Table 9.3 presents the overall performance of the YOLOv8l model in terms of evaluation metrics in both raw values and percentage formats. A brief comparison between the YOLOv8m and YOLOv8l models reveals that the larger model outperforms YOLOv8m in most key metrics, particularly in precision, recall, and F1-score, indicating that it provides

better overall performance in detecting fractures. While the mAP@50 is slightly lower in YOLOv8l, this is compensated by the fact that YOLOv8m achieves its optimal F1-score at a much lower confidence threshold of 0.254 compared to YOLOv8l, which is valued at 0.48. This is a crucial consideration as it corresponds to YOLOv8l producing more confidence predictions while requiring fewer low-confidence detection to achieve its best performance, suggesting that YOLOv8l is actually more selective and consistent in its detections. Additionally, YOLOv8l's considerably higher mAP@50-95 further supports the idea that it maintains better accuracy across a wider range of IoU thresholds, reinforcing its overall stability and generalization ability.

Table 9.3. Validation metrics for YOLOv8l

Metric	**YOLOv8l**	**YOLOv8l (%)**
Precision	0.8547	85.5%
Recall	0.5667	56.7%
mAP@50	0.498	49.8%
mAP@50-95	0.48	48.0%
F1-Score	0.6815	68.2%

9.3.2. YOLOv11 Detection Evaluation

In the case od the YOLOv11m model, the precision-confidence curve in Figure 9.22a indicates that the model maintains relatively low precision at lower confidence thresholds, meaning that many false positives are present when the model makes detections without high confidence. However, as the confidence threshold increases, precision improves steadily, indicating that higher-confidence predictions are more reliable. The curve reaches its peak at a confidence threshold of 0.784, where the model achieves maximum precision. This suggests that to minimize false positives, predictions should be made at a higher confidence level, reinforcing the idea that YOLOv11m benefits from a stricter decision boundary.

The precision-recall curve in Figure 9.22b highlights the trade-off between these two key metrics. As recall increases, precision begins to decline, following the expected inverse relationship. When the model tries to capture more true positives by increasing recall, it inadvertently introduces more false positives, reducing overall precision. The mAP@50 is

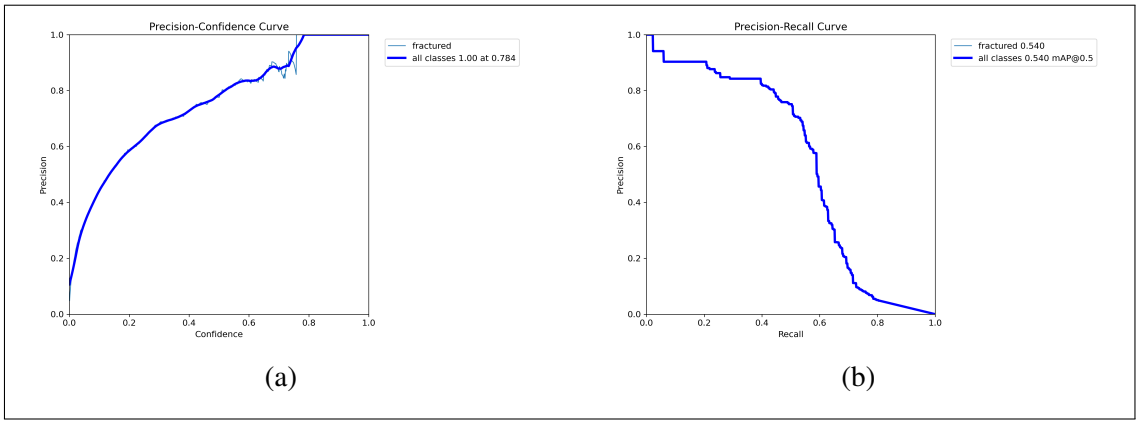


Figure 9.22. (a) YOLOv11m precision-confidence, (b) precision-recall curves on validation set.

0.54, meaning the model has moderate precision across different recall thresholds. The curve suggests that while the model is capable of high recall, maintaining both high precision and high recall simultaneously remains a challenge.

The recall-confidence curve presented in Figure 9.23a further supports this observation by highlighting how recall decreases as the confidence threshold increases. At very low confidence levels, recall is at its highest, meaning the model detects a larger number of fractures. However, this also includes many false positives, which is why precision is lower at those thresholds. As the confidence threshold increases recall steadily declines, eventually dropping off sharply at higher confidence values. This behavior suggests that while the model can detect many fractures, setting an excessively strict confidence threshold would lead to missed detections.

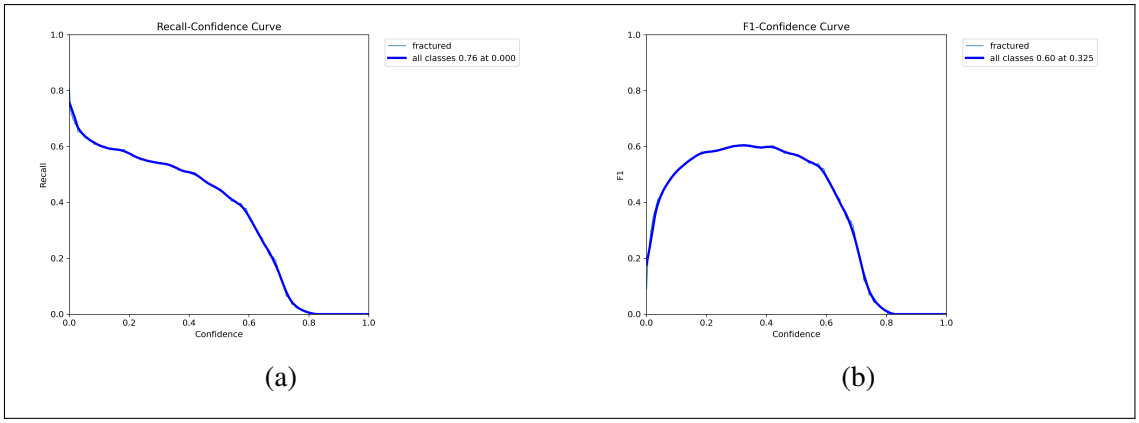


Figure 9.23. (a) YOLOv11m recall-confidence, (b) F1-Score-confidence curves on validation set.

The F1-Score across the confidence curve shown in Figure 9.23b provides a clearer indication of the optimal balance between precision and recall. The F1-score, which harmonizes these two metrics, peaks at a confidence threshold of 0.325, suggesting that at this level, the model achieves its best overall performance. Beyond this threshold, increasing confidence reduces recall too much, lowering the overall F1-Score. The maximum F1-score is 0.6, meaning that while the model is reasonably effective, it still struggles to perfectly balance precision and recall.

The normalized confusion matrix presented in Figure 9.24 for the YOLOv11m model provides insight into its detection performance. The model correctly predicts 82% of fractures, meaning that when it identifies a fracture, it is correct in most cases. However, 17% of predicted fractures are actually background, indicating that the model still generates some false positives. Even though it seems like 100% of the true background cases are misclassified as fractures, this is not true as the validation set does not contain any radiographs without any fractures in it.

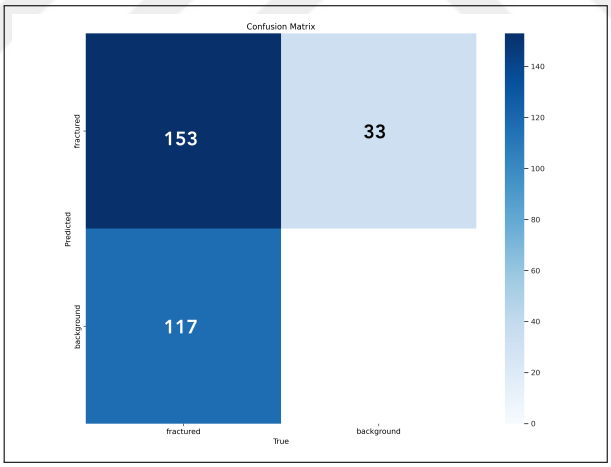


Figure 9.24. YOLOv11m normalized confusion matrix.

Overall, the YOLOv11m model demonstrates strong fracture detection capabilities, but it struggles with specificity. Table 9.4 shows the overall evaluation metrics for the trained model on the validation set. Consequently, this model's accuracy is 50.5% based on the metrics gathered from the confusion matrix.

For the evaluation of the last object detection algorithm, YOLOv11l, the precision-confidence curve is illustrated in Figure 9.25a. At lower confidence levels, precision is relatively low,

Table 9.4. Validation metrics for YOLOv11m

Metric	YOLOv11m	YOLOv11m (%)
Precision	0.8226	82.3%
Recall	0.5667	56.7%
mAP@50	0.54	54.0%
mAP@50-90	0.48	48.0%
F1-Score	0.6	60.0%

suggesting that the model includes many false positives when it makes predictions with lower certainty. However, as confidence increases, precision steadily improves, reaching its peak at 0.806 confidence, where the model achieves its most reliable fracture predictions. This indicates that YOLOv11l is more accurate when making high-confidence predictions, but at lower thresholds, it sacrifices precision for a higher number of detections.

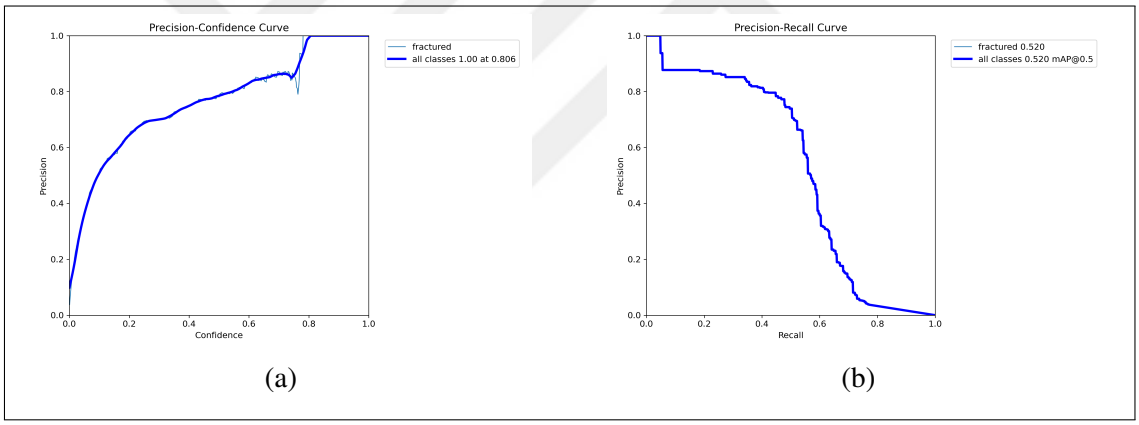


Figure 9.25. (a) YOLOv11l precision-confidence, (b) precision-recall curves on validation set.

The precision-recall curve in Figure 9.25b, on the other hand, highlights the inverse relationship between precision and recall. As recall increases, meaning the model captures more true fractures, precision begins to decline. The model reaches a mAP@50 of 0.520, showing that it maintains moderate precision across various recall levels. However, at higher recall values, precision drops significantly, indicating that detecting more fractures comes at the cost of including more false positives. The model needs to strike a balance between detecting as many fractures as possible while keeping false detections at a manageable level.

The recall confidence curve in Figure 9.26a further supports this observation by showing how recall declines as the confidence threshold increases. At very low confidence levels, recall is

at its highest, meaning the model detects almost all possible fractures. However, this comes at the expense of increased false positives. As the confidence threshold increases, recall gradually decreases, indicating that the model becomes more conservative in its predictions. At a confidence level beyond 0.7, recall drops sharply, meaning that setting the threshold too high would result in many missed fractures. This suggests that while a high confidence threshold improves precision, it also leads to a loss in sensitivity.

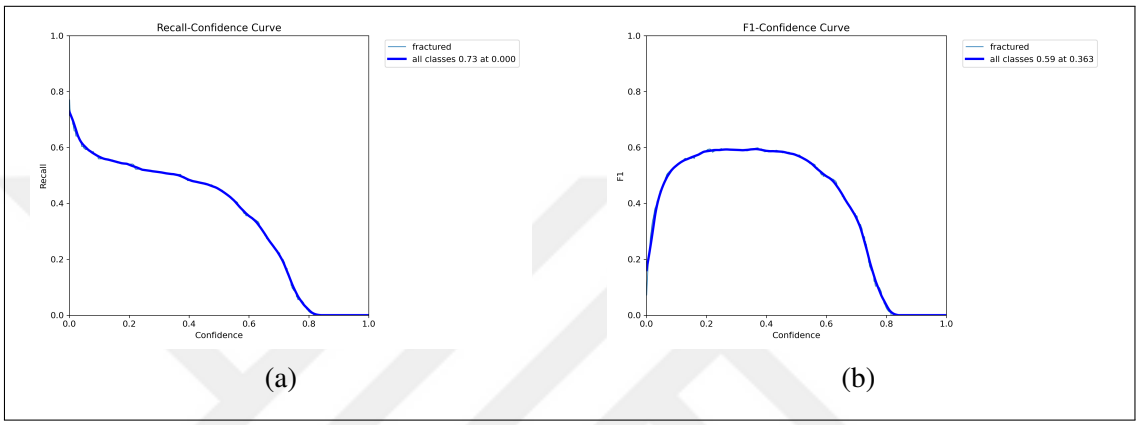


Figure 9.26. (a) YOLOv11m recall-confidence, (b) F1-Score-confidence curves on validation set.

The F1-confidence curve presented in Figure 9.26b helps identify the best balance between precision and recall. The F1-score, which considers both metrics, reaches its peak at a confidence threshold of 0.363, suggesting that this is the optimal level for balancing the trade-off between correctly identifying fractures and minimizing false positives. The maximum F1-score achieved is 0.59, indicating that the model provides a reasonable balance.

Thus, these results show that YOLOv11l performs best when confidence thresholds are carefully tuned. The optimal confidence level of 0.363 for the F1-Score suggests that the model requires a moderately strict threshold to achieve the best trade-off. While high-confidence predictions are highly precise, the recall trade-off means that missing fractures is a potential risk. The model's mAP@50 of 0.52 indicates it has reliability across different recall levels.

The normalized confusion matrix for the YOLOv11l model is presented in Figure 9.27, providing insight into its predictive behavior, particularly in the absence of true negative instances in the dataset. The model correctly identifies 90% of fractures, meaning that when

it predicts a fracture, it is correct the vast majority of the time. However, 10% of predicted fractures are actually background, which indicates that while the model has strong detection capabilities, it still produces some false positives.

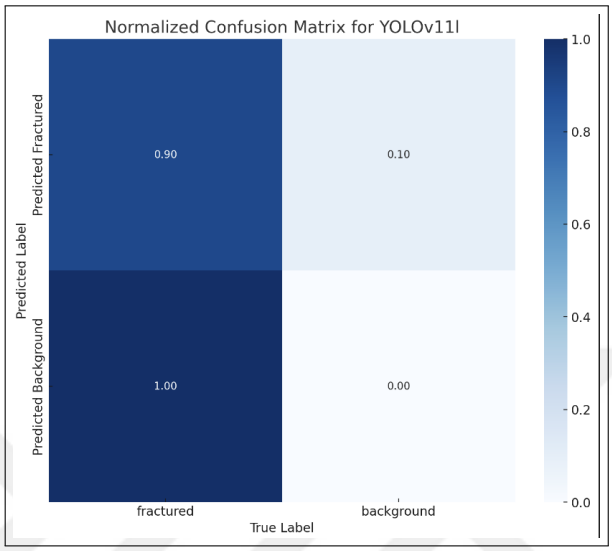


Figure 9.27. YOLOv11l normalized confusion matrix.

A key observation is that 100% of the actual background instances are classified as fractures, but this does not reflect an error in the model's performance. Instead, it is a direct consequence of the dataset composition, which does not contain any true background-only cases. Overall, the YOLOv11l model demonstrates a high level of confidence in detecting fractures, with over 90% accuracy in fracture prediction.

Table 9.5. Validation metrics for YOLOv11l

Metric	**YOLOv11l**	**YOLOv11l (%)**
Precision	0.9036	90.4%
Recall	0.5667	56.7%
mAP@50	0.52	52.0%
mAP@50-90	0.48	48.0%
F1-Score	0.59	59.0%

Finally, Table 9.5 shows the overall metrics gathered through the validation set for the YOLOv11l model. A brief comparison between YOLOv11m and YOLOv11l models reveals that the larger model achieves a higher precision while maintaining the same recall. Furthermore, despite both models achieving the same mAP@50-90, the mAP@50 is slightly

higher for YOLOv11m with a faintly higher F1-Score. This means that YOLOv11l offers better precision, making it more reliable in correctly identifying fractures, however, the YOLOv11m model maintains a bit better balance between precision and recall.

9.4. YOLO DETECTION REAL-WORLD TESTING

After evaluating the YOLO models on the validation set, it is crucial to assess their performance on real-world test data to determine their generalizability and practical usability. While validation results provide insights into how well the models learn from the training data, real-world testing reveals how they handle new cases, including variations in image quality, anatomical differences, and challenging fracture patterns. In this section, the predictions made by the YOLO models on the test set are analyzed, highlighting both successful detections and erroneous classifications. Special attention is given to false positives and false negatives, exploring potential reasons for these misclassifications, such as the presence of subtle fractures, overlapping structures, or model biases. This analysis serves as a final evaluation of the YOLOv8m, YOLOv8l, YOLOv11m, and YOLOv11l models' robustness, helping to determine their reliability for practical deployment in fracture detection.

In the real-world test set used for evaluating the YOLO models, there is a total of 100 test images, identical to the Faster R-CNN testing procedure, with 116 ground truth fracture annotations. This is because certain images contain more than one fracture in a single wrist, while others include two different views of the same wrist within a single radiograph. As a result, the test set presents a more complex detection scenario, requiring the models not only to identify fractures but also to accurately localize multiple instances within a single image. This setup mirrors real-world clinical challenges, where radiographs often capture multiple anatomical perspectives or overlapping fractures, making detection more intricate.

For the testing procedure of YOLOv8m, two different confidence thresholds are considered to evaluate the model's performance. The first threshold, 0.25, is derived from the F1-confidence curve, indicating the optimal balance between precision and recall. The second threshold, 0.5, is a commonly used default value in object detection tasks. However,

evaluation results -particularly precision-recall performance and qualitative inspection of sample predictions— indicated that the model produced more accurate and reliable results using the 0.25 confidence threshold. This setting captured more fractures while maintaining a reasonable level of precision. As a result, all final predictions for YOLOv8m were made at this lower confidence threshold, ensuring that the model's detection capabilities were maximized for real-world testing.

During the real-world testing of YOLOv8m, the model produced 8 false positive detections out of the 116 ground truth fractures. A closer analysis of these misclassifications revealed the following three primary causes: (i) in 4 instances, the false positives resulted from extra bounding boxes either appearing within an already correctly predicted fracture or being placed immediately adjacent to an accurate detection. These redundant bounding boxes suggest that the model may sometimes struggle with bounding box refinement, potentially over-segmenting fractures, or detecting minor variations within a single fracture as separate entities.

(ii) In 2 cases, the model predicted fractures in areas where no actual fracture was present, leading to genuine false positive detections. These errors indicate that the model may have misinterpreted anatomical structures, noise, or other radiographic irregularities as fractures. (iii) The remaining 2 false positives were due to incorrect bounding box placements, where the model identified a fracture but did not accurately position the detection box over the actual fracture location. This suggests that while the model recognizes fracture-like patterns, it may occasionally misalign its predictions, leading to detection errors that could affect clinical usability.

On the other hand, the model failed to detect 8 fractures, resulting in false negatives. Out of these 8 instances, in 5 of them, the false negatives can be attributed to the subtlety of the fractures, meaning that the fractures were either faint, had minimal displacement, or blended into surrounding bone structures, making them difficult for the model to distinguish. These cases highlight the challenge of detecting subtle fractures, a known difficulty even for human radiologists.

The other false negative occurred in a case where the patient had a cast, which likely obscured part of the fracture or introduced additional structural elements that confused the model. The

presence of foreign materials in radiographs can affect detection performance, as models are primarily trained on unobstructed fracture images. Another false negative involved an image containing both AP and lateral views of the wrist within a single radiograph. While the model successfully detected the fracture in the AP view, it failed to identify the fracture in the lateral view. This suggests that the model may perform better on certain viewing angles or that the overlapping nature of two projections in one image increased the difficulty of detection.

The final false negative was unexplained, meaning that there was no obvious justification for why the model missed the fracture. The fracture was visible, and the surrounding conditions did not appear significantly different from correctly detected cases. This suggests that some inherent variability in the model's detection patterns could lead to occasional unpredictable failures.

It is also important to highlight that YOLOv8m has an improved ability to detect complex fractures compared to Faster R-CNN. The fact that 3 out of 5 casted radiographs missed by Faster R-CNN are successfully identified by YOLOv8m suggests that the model is better equipped to handle images with external obstructions, such as orthopedic casts, which often introduce visual challenges. Additionally, a particularly difficult two-view radiograph with low contrast and a cast, which was a false negative for Faster R-CNN, is correctly identified by YOLOv8m, as presented in Figure 9.28a. This suggests that YOLOv8m is more resilient to low contrast related conditions, overlapping structures, and obstructed regions, making it more suitable for real-world clinical applications where image quality can vary significantly.

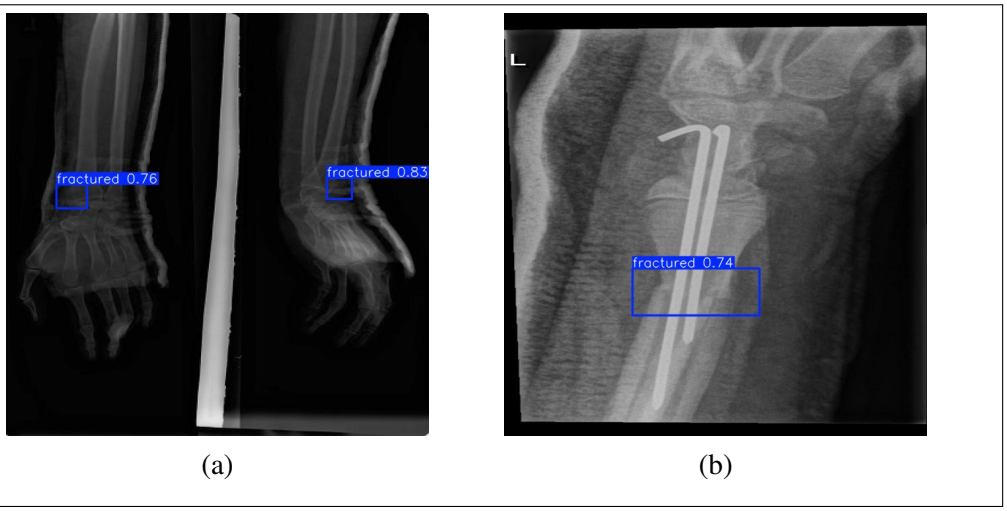


Figure 9.28. (a) YOLOv8m cast and (b) surgical hardware TP predictions.

Similarly, the 2 cases entailing surgical hardware are correctly detected by YOLOv8m, whereas were failed with Faster R-CNN. This improvement indicates that YOLOv8m is less sensitive to artifacts caused by metal implants, which can sometimes interfere with fracture visibility in X-ray images. Figure 9.28b shows the prediction on the image from the test set with the surgical screw missed by the Faster R-CNN model.

The confusion matrix with raw values is presented in Figure 9.29, summarizing the findings on the test set. The results indicate that 93.1% of actual fractures are correctly detected, meaning the model has a strong ability to identify fractures. However, 6.9% of the predicted fractures were false positives, where the model incorrectly identified a fracture that did not exist or placed extra bounding boxes.

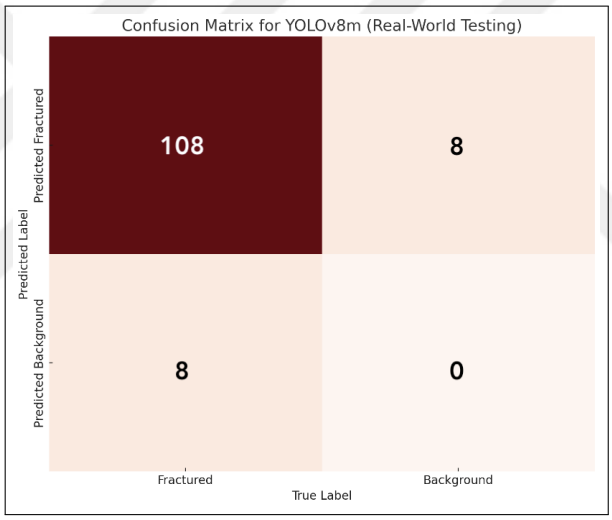


Figure 9.29. YOLOv8m raw confusion matrix on the test set.

Based on the findings from the confusion matrix, the other evaluation metrics can be calculated. The metrics computed on the test set are shown in Table 9.6. The precision and recall are identical as there is an even number of FPs and FNs, thus, the F1-Score is the same as well. Finally, the accuracy of 87.1% represents the overall effectiveness of the model in making correct predictions relative to all detections made. While this is a strong performance, the fact that some false positives and false negatives still occur suggests that there is room for further improvement.

For the YOLOv8l testing, the F1-Score-confidence curve suggested an optimal confidence threshold of 0.5, which coincides with the commonly used default threshold. As a result, all

Table 9.6. Performance metrics for YOLOv8m on the test set

Metric	**YOLOv8m**	**YOLOv8m (%)**
Precision	0.931	93.1%
Recall	0.931	93.1%
F1-Score	0.931	93.1%
Accuracy	0.870	87.1%

test predictions for YOLOv8l are conducted at a confidence threshold of 0.5. This threshold is selected to ensure a balance between precision and recall, optimizing the model's ability to detect fractures while minimizing false detections.

Consequently, for the testing of the YOLOv8l model, the same testing set is utilized. The model exhibited 6 false positives and 9 false negatives. The false positive cases were identical to those observed in YOLOv8m, meaning the same errors related to extra bounding boxes, mislocalized predictions, and incorrect fracture detections persisted in the larger model, however, 2 of the mis-detections were resolved. Thus, it can be stated that YOLOv8l does not introduce new false positive patterns. Nonetheless, the false negative count increased from 8 in YOLOv8m to 9 in YOLOv8l, indicating that the larger model missed one additional fracture. The 7 false negatives observed in YOLOv8m were also present in YOLOv8l, suggesting that these errors were particularly challenging for both models. The remaining two false negatives were due to an image containing a two-view radiograph where both views were missed.

The raw confusion matrix for the YOLOv8l model testing is presented in Figure 9.30. The matrix reveals that the model identified 107 fractures, but it also produced 6 false positives, meaning it predicted a fracture where none existed. Additionally, 9 fractures were missed, resulting in false negatives. Compared to YOLOv8m, YOLOv8l exhibits a slightly higher false negative rate, meaning it failed to detect more fractures. The 6 false positives were identical to those in YOLOv8m, suggesting that both models share similar error patterns when it comes to mistakenly identifying non-fractured areas as fractures. However, YOLOv8l struggled more with two-view radiographs, as it failed to detect both fractures in a case where two perspectives were present in a single image.

However, it is crucial to highlight that YOLOv8l's confidence threshold is set to 0.5 whereas

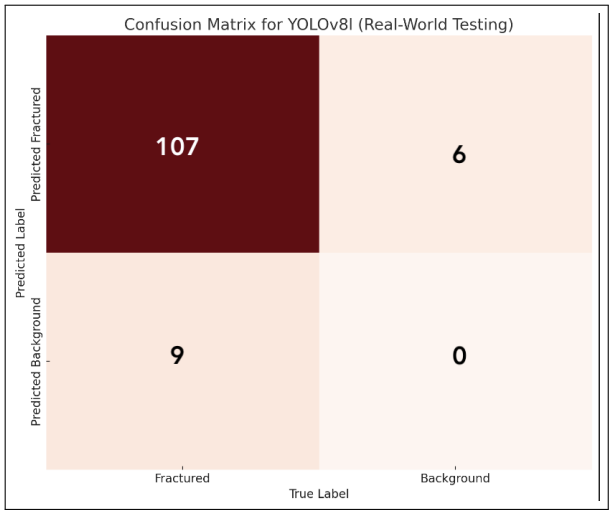


Figure 9.30. YOLOv8l raw confusion matrix on the test set.

the YOLOv8m's is 0.25. This corresponds to YOLOv8l making predictions with a much higher level of certainty, which directly influences its higher precision and lower false positive rate, as it only predicts fractures when it is more confident. While making predictions, YOLOv8l is approximately twice as confident in its predictions compared to YOLOv8m, which explains why it has fewer false positives but slightly more false negatives. The model is more selective in making predictions, preferring to avoid false detections at the cost of missing some fractures. This trade-off is expected when increasing the confidence threshold, as higher confidence leads to better precision but can sometimes reduce recall. This aspect is crucial for real-world applicability. In clinical settings, a model like YOLOv8l, which is highly confident in its detections, might be preferred to reduce unnecessary false alarms.

In Table 9.7, the performance metrics for the YOLOv8l model are displayed. The metrics are derived through the confusion matrix. The YOLOv8's larger model exhibits a higher precision of 94.7% compared to the medium model, which suggests that when the model predicts a fracture, it is correct in nearly all cases, meaning false positives are minimal. The recall, on the other hand, is slightly lower compared to the medium model.

The F1-score of 93.4% confirms that YOLOv8l achieves an ideal balance between precision and recall, making it a more reliable option for fracture detection. Despite the slight drop in recall, the high precision helps maintain a strong overall performance, exceeding the F1-Score of the medium model. The accuracy of 87.7% represents the overall effectiveness of the model

Table 9.7. YOLOv8m performance metrics on test set

Metric	**YOLOv8l**	**YOLOv8l (%)**
Precision	0.946	94.7%
Recall	0.922	92.2%
F1-Score	0.934	93.4%
Accuracy	0.877	87.7%

in making correct predictions relative to all detections made. This metric is slightly higher than YOLOv8m, meaning the model makes fewer incorrect predictions overall.

For the testing of YOLOv11m, the confidence threshold of 0.3 is selected as the F1-Score across the confidence curve suggests for the predictions on the test set, with the goal of maximizing detection accuracy while minimizing false predictions. The test set remained the same, consisting of 100 test images containing a total of 116 ground truth fractures. The results of the YOLOv11m testing reveal that the model produced 5 false positives in total, however, a closer analysis of these instances disclose that not all of them represent entirely incorrect detections.

One of these false positives was correctly classified in the YOLOv8m and YOLOv8l tests but an additional bounding box in YOLOv11m is produced. This suggests that the model may have slightly different sensitivity to certain fracture patterns compared to its predecessors. Two other false positives are identical to those observed in YOLOv8, meaning that these misclassifications persisted across multiple YOLO models, likely indicating challenging cases where non-fractured regions exhibit features that resemble fractures.

The final two false positives, however, are not entirely incorrect in a clinical sense. These cases involved severe fracture fragmentation, where the model placed multiple bounding boxes over different parts of the fracture instead of a single detection, as exhibited in Figure 9.31. Given that these types of fractures are complex, with bone fragments being separated from one another, the model detected multiple regions as potential fractures. While this technically results in a false positive count, the model still successfully identified the fractured regions, it simply over-segmented them due to the spatial separation of bone fragments.

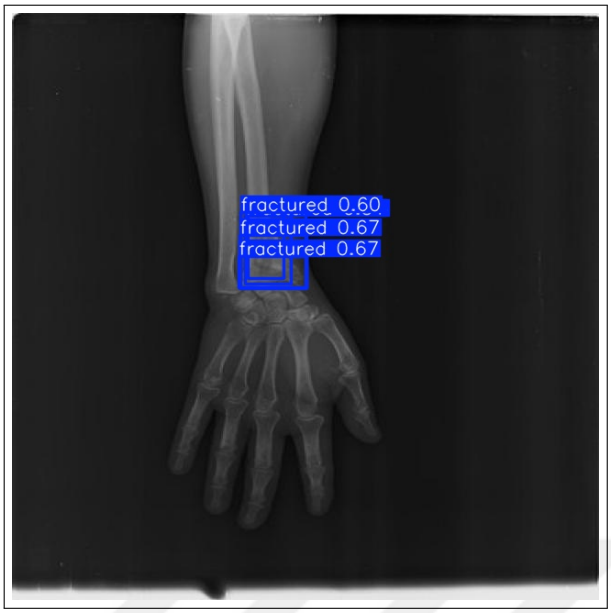


Figure 9.31. YOLOv11m prediction of multiple bounding boxes.

Another particularly notable instance from the test set, presented in Figure 9.32, highlights a key improvement in YOLOv11m's localization capabilities compared to its predecessors. In both YOLOv8m and YOLOv8l, the bounding box placement did not correctly correspond to the actual fracture, as shown in Figure 9.32a, leading to a false positive, as the detected area did not contain a fracture. However, in the same case, YOLOv11m successfully identified and localized the fracture correctly, even in the presence of a cast can be seen in Figure 9.32b.

In the YOLOv11m real-world testing, the model produced a total of 8 false negatives, with most cases aligning with the errors observed in the YOLOv8 models. Particularly, 6 of these false negatives are the same as those in YOLOv8m and YOLOv8l, indicating that these fractures were particularly challenging across different model versions. However, one false negative stands out as an exceptional case as it was correctly detected in both YOLOv8m and YOLOv8l but was missed by YOLOv11m. This suggests that while YOLOv11m demonstrates improvements in localization and certain challenging conditions, it might potentially have slightly different sensitivity patterns, leading to occasional missed detections that were previously captured.

Additionally, in Figure 9.33, an image containing two casted fractures with low contrast that was previously discussed is partially undetected. While the model successfully detected one of the fractures, it missed the second, which may indicate that casted, low-contrast fractures

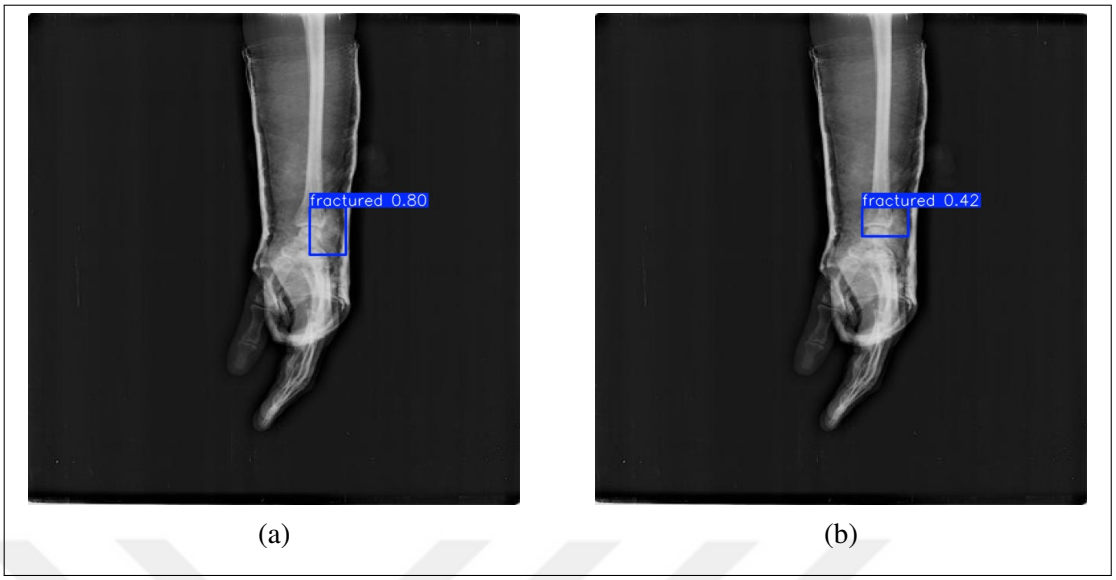


Figure 9.32. (a) Prediction of YOLOv8 models (b) versus the YOLOv11m model on the same image.

remain a challenge for YOLOv11m, particularly in cases where fractures are less visually distinct.



Figure 9.33. YOLOv11m prediction on the casted, low constrast test instance.

The raw confusion matrix for YOLOv11m is presented in Figure 9.34, delivering a clear representation of the model's performance. The model correctly identified 108 fractures, while 5 false positives were recorded. Additionally, 8 fractures were missed, leading to false negatives. Compared to previous YOLO models, YOLOv11m exhibits a lower false positive count, indicating that it is more precise in its fracture predictions. This suggests that YOLOv11m is better at avoiding unnecessary detections in non-fractured regions. However,

its false negatives remain the same as in YOLOv8m, meaning that certain challenging cases continue to pose detection difficulties.

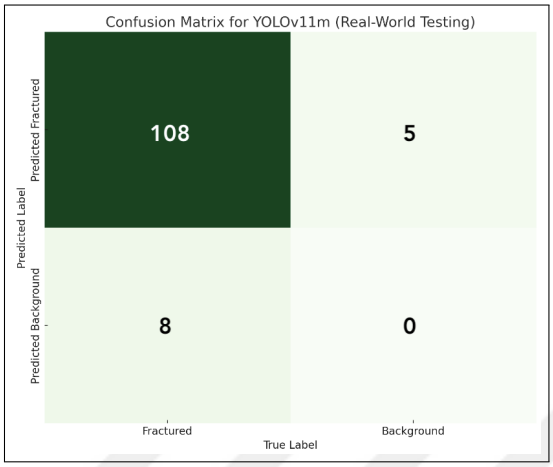


Figure 9.34. YOLOv11m raw confusion matrix on the test set.

Notably, the reduction in false positives may be linked to the lower confidence threshold used for predictions. By setting a confidence threshold that allows slightly less strict detections, the model maintains strong recall while improving its overall precision. Additionally, cases where multiple bounding boxes were drawn due to fracture fragmentation suggest that YOLOv11m has a more detailed understanding of fractures, even if it sometimes over-segments them. Overall, YOLOv11m demonstrates an improvement in precision over YOLOv8 models while maintaining a similar recall rate, as presented in Table 9.8.

Table 9.8. YOLOv11m performance metrics on test set

Metric	**YOLOv11m**	**YOLOv11m (%)**
Precision	0.955	95.6%
Recall	0.931	93.1%
F1-Score	0.943	94.3%
Accuracy	0.892	89.3%

The precision of 95.6% of the YOLOv11m model indicates that when it predicts a fracture, it is correct in nearly all cases, marking an improvement over the previous YOLO models, showing that the model produces even fewer false positives while maintaining strong detection capabilities. The recall of 93.1%, on the other hand, confirms that the model successfully identifies the majority of fractures. Despite the few additional false negatives, YOLOv11m

still maintains strong sensitivity, ensuring that most actual fractures are detected. The F1-score of 94.3% displays an adequate balance between precision and recall, confirming that YOLOv11m provides both high detection accuracy and minimal false detections. Finally, the accuracy of 89.3% is the highest among all tested YOLO models so far, demonstrating that YOLOv11m is more reliable in making correct predictions overall.

Lastly, for the YOLOv11l testing, the F1-Score-Confidence curve suggested a confidence threshold of 0.36, which is used for all test predictions. The test set remained the same and consistent, which has 100 test images with a total of 116 ground truth fractures.

The YOLOv11l produced 5 false negatives and 4 false positives, but what stands out is that all of these errors are the same as those in YOLOv11m, but slightly improved. This suggests that YOLOv11l refines the detection capabilities of YOLOv11m, reducing both missed and incorrectly predicted fractures. Remarkably, one fracture that was missed in YOLOv11m but correctly detected in YOLOv8m and YOLOv8l was also correctly detected in YOLOv11l. This suggests that YOLOv11l retains some of the strengths of the earlier YOLOv8 models while improving upon the overall precision and recall of YOLOv11m. These findings indicate that YOLOv11l achieves a more balanced and refined detection performance.

Table 9.9. YOLOv11l performance metrics on test set

Metric	**YOLOv11l**	**YOLOv11l (%)**
Precision	0.965	96.5%
Recall	0.956	95.7%
F1-Score	0.961	96.1%
Accuracy	0.925	92.5%

Based on the confusion matrix, the remaining metrics are calculated and presented in Table 9.9. The overall metrics indicate that the YOLOv11l model has the highest precision among all tested YOLO models, demonstrating reliable detections with minimal false positives. The recall value of 95.7% is likewise the highest recall achieved thus far and compared to previous YOLO models, highlights an improvement in sensitivity. Similarly, the F1-Score further confirms that the YOLOv11l model surpasses the previous models. The accuracy of 92.5% is the highest accuracy value observed out of all detection models, proving that it makes the fewest overall incorrect predictions on the test process.

9.5. COMPARISON AND ASSESSMENT OF DETECTION MODELS

To determine the best-performing detection model, multiple factors are considered, including detection accuracy, error distribution, and inference time. Precision, recall, F1-score, and overall accuracy provide a clear view of each model's ability to detect fractures while minimizing false predictions. Among the evaluated models (Faster R-CNN, YOLOv8m, YOLOv8l, YOLOv11m, and YOLOv11l), Faster R-CNN consistently underperforms across all metrics, as observed in Section 9.1 and Section 9.2. As a result, it is excluded from further comparisons, being the first model to be eliminated.

Among the YOLO models, YOLOv11l consistently achieved the highest precision, recall, and F1-Score, making it the most balanced detector in terms of false positives and false negatives. YOLOv8 models had a lower recall, missing more fractures, while YOLOv11m showed improvements but still exhibited slight inconsistencies in certain cases where previous models had succeeded. A side-by-side performance comparison presented in Table 9.10 reveals that the YOLOv11l model exhibits the highest precision and recall, reducing both FNs and FPs. The YOLOv11m model, on the other hand, performed similarly to the larger model, however, it had a slightly lower recall. The YOLOv8 models exhibited more false negatives, particularly in complex radiographs.

Table 9.10. Comparison of YOLO detection models on the test set

Metric	**YOLOv8m**	**YOLOv8l**	**YOLOv11m**	**YOLOv11l**
Precision	93.1%	94.7%	95.6%	96.5%
Recall	93.1%	92.2%	93.1%	95.7%
F1-Score	93.1%	93.4%	94.3%	96.1%
Accuracy	87.1%	87.7%	89.3%	92.5%

A consolidated analysis of false positives and false negatives across all models highlights some common trends. Fractures with subtle visual indicators are the most frequent false negatives across all models. Casted fractures and two-view radiographs were particularly challenging for Faster R-CNN but were progressively handled better in YOLO models, particularly YOLOv11l. Surgical hardware presented difficulties for Faster R-CNN but was handled well by YOLO models. FPs were often due to either bounding box placement errors (misalignment or redundancy) or over-segmentation of fragmented

fractures. Nevertheless, YOLOv11l demonstrated the most refined localization, correctly detecting fractures previously misclassified in YOLOv11m.

The average inference time per image is shown across models in Table 9.11, revealing that the YOLOv8m is the fastest model whereas the YOLOv11l model is the slowest due to its more complex architecture.

Table 9.11. Comparison of inference times of YOLO detection models

Metric	**YOLOv8m**	**YOLOv8l**	**YOLOv11m**	**YOLOv11l**
Inference Time (ms)	9.8	11.9	12.4	18.3

As expected, the YOLOv11l model has a cost factor of longer processing time, however, its metrics compensate for the inference time. Nonetheless, in a medical environment, the primary concern when selecting an object detection model would be its accuracy rather than its speed, especially when the time difference between models is in milliseconds. Unlike real-time applications such as autonomous driving, where rapid decision-making is critical, fracture detection in radiographs is a high-stakes diagnostic task where misdiagnoses can have serious consequences. The most significant risk in medical imaging is FNs, as these can lead to delayed treatment, mismanagement of injuries, and potential complications for the patient. While false positives are undesirable because they may lead to unnecessary further examinations or interventions, they are generally less critical than failing to identify an actual fracture. YOLOv11l minimizes both FPs and FNs better than any other tested model, making it the most reliable option for a clinical setting.

Furthermore, even the slowest model, YOLOv11l with 18.3 ms inference time per image, is still extremely fast compared to the time taken for a radiologist to manually assess an image. The difference between YOLOv8m and YOLOv11l is less than 10 milliseconds; an insignificant delay in a workflow where normally a radiologist spends several minutes evaluating a set of scans. Therefore, prioritizing accuracy over speed is the most logical choice, as even the most accurate model remains computationally efficient for practical clinical use. Based on these findings, YOLOv11l stands out as the best detection model for fracture identification in radiographs, offering the highest accuracy in a reasonable inference time.

10. EVALUATION AND TESTING OF THE SEGMENTATION MODELS

The flowchart illustrated in Figure 10.1 shows the methodolody followed in evaluating and testing Mask R-CNN, EfficientNet and YOLO segmentation models. The process is divided into three main phases, namely overview, model-specific assessment, and final comparison. The definition phase consists of overviewing the methodology utilized for evaluating the segmentation models and providing clarity before diving into specific evaluations.

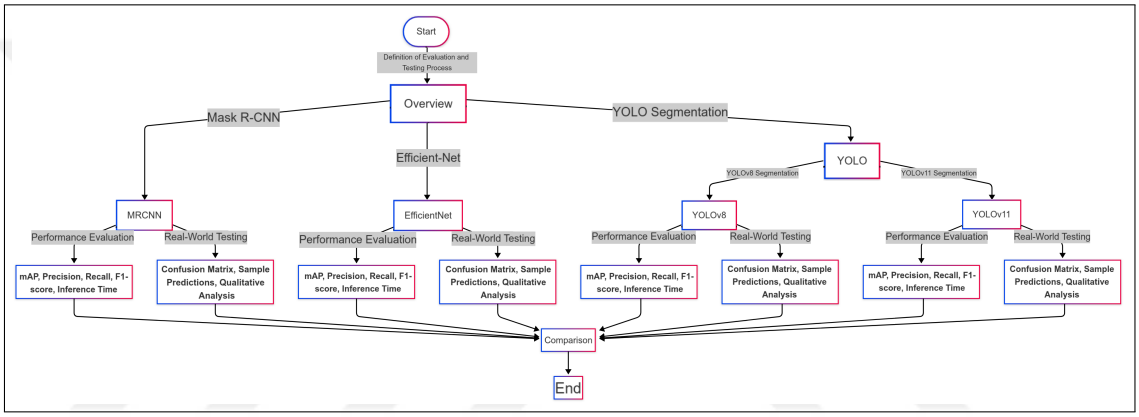


Figure 10.1. Flowchart of the procedure for evaluating and testing the segmentation models.

In model-specific assessment, Mask R-CNN, Efficient-Net, and YOLO-based segmentation models are individually evaluated and tested. Identical to the detection evaluation and testing, each model undergoes two key phases; performance evaluation on the validation set and real-world testing. The performance evaluation procedure utilizes quantitive metrics such as mAP, precision, recall, F1-score, and inference times to analyze the models. Real-world testing, on the other hand, involves testing the models on a separate dataset to assess their ability to generalize. This second phase is evaluated using confusion matrices, sample predictions, and quantitive analysis to examine how well the models identify fractures in new, unseen radiographs. Finally, the comparison part consists of synthesizing the evaluation and testing results to compare models and identify the best-performing segmentation approach based on various metrics.

10.1. MASK R-CNN PERFORMANCE EVALUATION

Mask R-CNN is a widely recognized segmentation model and this section provides a comprehensive evaluation of the model's performance using both training metrics and validation results to assess its effectiveness in segmenting DRFs.

The validation set is utilized to compute segmentation metrics, including mask average precision at different thresholds, recall, and F1 score to examine the detection performance of the model appropriately. As mentioned previously in Chapter 7, the training is conducted for 8,000 iterations with validation metrics computed at every 500th iteration, resulting in a total of 16 evaluation steps. Furthermore, the model is saved at every evaluation checkpoint to utilize the best-performing model in the testing section later on.

The curve for mask loss for the Mask R-CNN model is exhibited in Figure 10.2 on the validation set. From the curve, it can be observed that the model starts with an initially high loss. As expected, the loss drops rapidly during the first 1,000 iterations. Thereafter, the loss starts to decrease gradually and fluctuates between 0.4 and 0.45 from the 1,000th iteration to the 3,000th iteration. These fluctuations indicate that the model is still adjusting but has learned basic segmentation. Around approximately 3,500 iterations, the loss stops decreasing, fluctuating around 0.38 suggesting that the model is no longer improving significantly.

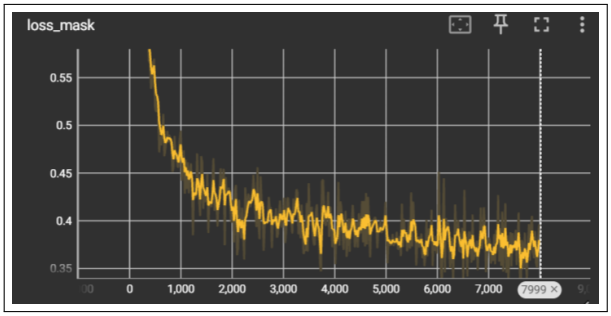


Figure 10.2. Mask R-CNN mask loss curve during training.

Another metric computed on the validation set during training is accuracy curve. Hence, Figure 10.3 presents the mask accuracy of the Mask R-CNN model over the training iterations, computed based on the predictions made on the validation set. Interpretation of the curve reveals that there is a steady increase in accuracy within the initial 2,000 iterations, which

shows that the model is learning effectively in the early stages. Accuracy continues to improve at a slower rate until the 4,000th iteration, reaching approximately 0.8 accuracy. Afterward, accuracy fluctuates and reaches 0.83 at the maximum, suggesting the model has mostly converged and is no longer learning significantly.

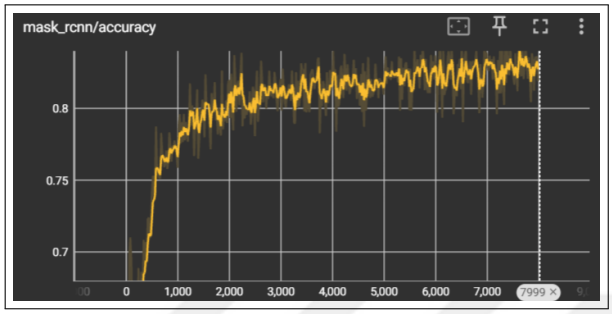


Figure 10.3. Mask R-CNN mask accuracy curve during training.

The accuracy curve shown in Figure 10.2 complements the loss curve presented in Figure 10.3 as the fluctuations and plateaus occur around the same iterations, confirming that further training might not be improving the model significantly. The graph of FNs on the validation set shown in Figure 10.4 tells a different story. As expected, the model exhibits some initial instability, with the FN rate fluctuating a lot up until the 1,000th iteration, which can be attributed to random initialization. It spikes up above 0.14, meaning the model is missing many actual fractures at the beginning. After 1000 iterations, the false negative rate drops to approximately 0.12. Thereafter, the model does not show a strong downward trend, meaning the model is not improving much in terms of reducing missed fractures.

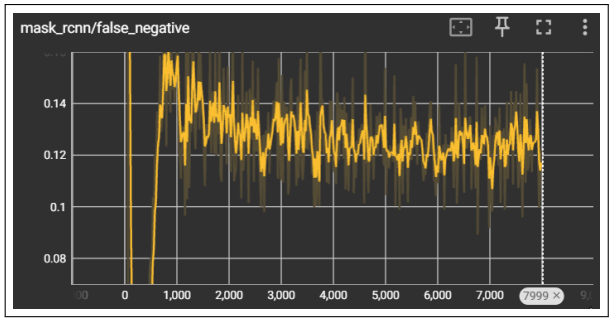


Figure 10.4. Mask R-CNN FN rate curve during training.

Similarly, in Figure 10.5, the FP rate over training iterations is presented. The false positive rate starts very high at the beginning, which is expected since the model is making a lot of incorrect predictions early in training. The false positive rate rapidly drops and stabilizes

around 0.3 by 1,000 iterations, suggesting the model is learning to avoid unnecessary detections. Thereafter, the FP rate starts to decrease more slowly, dropping to 0.19 between the 7,000th and 8,000th iterations, indicating that misclassifications might still happen, however, the model has become more precise compared to the beginning of the training.

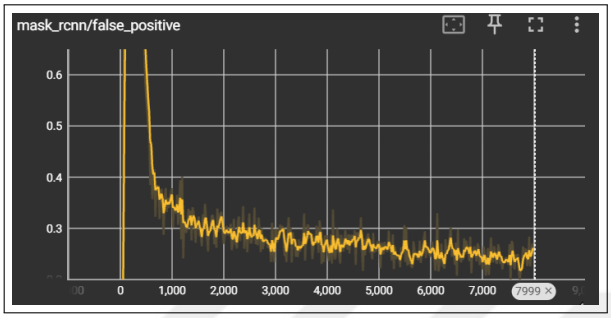
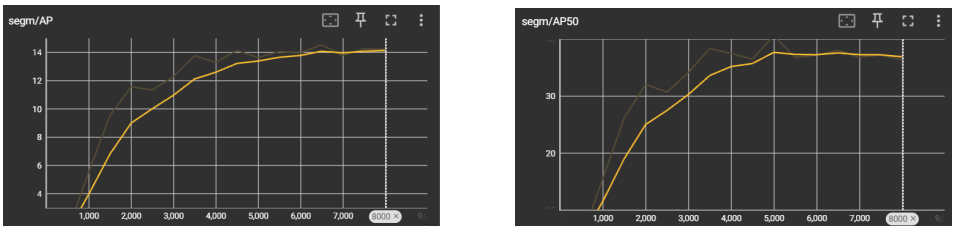


Figure 10.5. Mask R-CNN FP rate curve during training.

Last but not least, Figure 10.6 shows the segmentation performance of the Mask R-CNN model over training iterations, with Figure 10.6a presenting the AP at various IoU thresholds and Figure 10.6b at the fixed IoU threshold of 50. The mAP@0.50:0.95 shows a rapid increase until the end of the training with a slower growth between the 3,000th and 6,000th iterations, reaching a maximum AP of 14.25%, which is expected since fractures are such small things to segment at stricter confidence thresholds. The curve in Figure 10.6b shows a similar trend where the AP increments rapidly, reaching a maximum of 40.6% AP@50 on the validation set. Thus, it can be concluded that model performance is improving and stabilizing throughout the training, however, no major improvements are observed after approximately 6,500 iterations.



(a) (b)

Figure 10.6. (a) Mask R-CNN AP at IoU 0.50:0.95, (b) AP at IoU threshold set to 50.

InFigure 10.7, the confusion matrix for the validation set from the best-performing iteration based on the metrics is presented. Accordingly, it can be concluded that the model correctly

identifies a majority of fractures, which means the model is reasonably reliable for fracture segmentation. However, the model often detects fractures where there are none, particularly in a quarter of the validation images, which may lead to false alarms. Furthermore, the FN count reveals that the model fails to detect some fractures, meaning it may not be sensitive enough, prioritizing sensitivity over precision. Nonetheless, this will be aimed to be solved by fine-tuning the confidence threshold while testing with additional data.

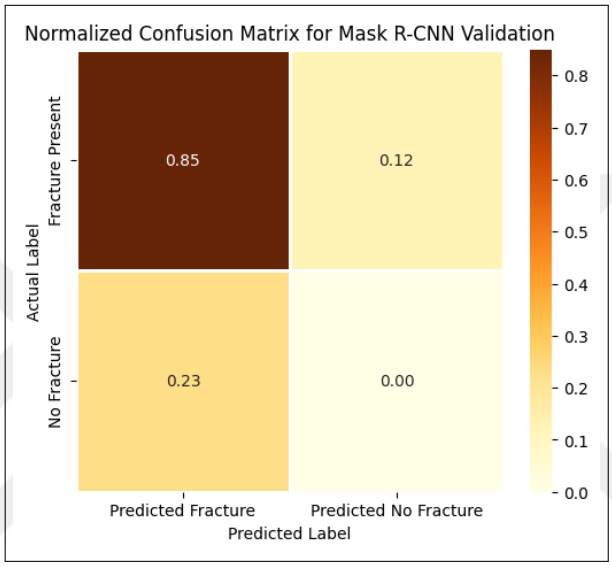


Figure 10.7. Mask R-CNN confusion matrix on the validation set.

Yet, it should be heared in mind that in a medical environment, sensitivity is far more important than specificity, since a FN could have serious consequences for the patient's treatment. FPs are less harmful, as they may lead to additional medical review, however, this is acceptable as a radiologist is required to verify the result nonetheless, leading to an extra X-ray in worst cases. FNs, on the other hand, are more risky, as missing an actual fracture might mean delayed treatment and may lead to misalignment, improper healing, or long-term disability.

10.2. MASK R-CNN REAL-WORLD TESTING

After the validation step, the trained Mask R-CNN model is evaluated on an independent test set consisting of 100 images and 102 radiographic ground truths. This evaluation aims to assess the model's effectiveness in detecting and segmenting fractures under real-world

conditions. On the test set, the model achieved 84 TPs, 13 FPs, and 5 FNs. These results complement the findings obtained via the validation set, maintaining high sensitivity while keeping false negatives to a minimum. Figure 10.8 shows the confusion matrix of the Mask R-CNN.

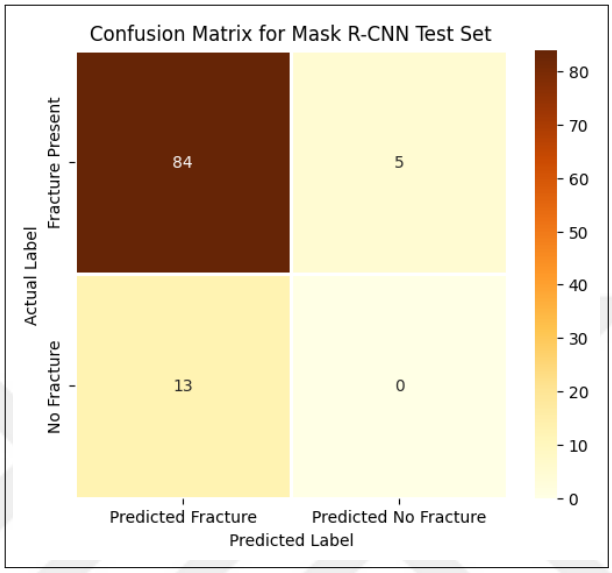


Figure 10.8. Mask R-CNN confusion matrix on the test set.

During the evaluation of the Mask R-CNN segmentation model on the test set, orthopedic surgeons identified a few noteworthy cases that provide insight into the model's behavior in complex scenarios. One such case involved a radiograph containing two separate wrist fractures, both of which were successfully detected by the model. However, in another case, the model misclassified a bracelet on the patient's wrist as an additional fracture, resulting in a false positive detection, as shown in Figure 10.9. Despite the additional false prediction, the ability of the model to detect both fractures correctly suggests that it generalizes well across multiple fractures within a single radiograph.

However, the misclassification of the bracelet as a fracture indicates that the model may be sensitive to irregular structures that resemble fracture patterns in X-rays. In X-ray images, metallic objects, such as jewelry, surgical implants, or medical hardware, appear as high-contrast regions, similar to fracture lines. The overall model design prioritizes sensitivity, aiming to detect as many fractures as possible while allowing some false positives, however, in this case, the model successfully detected both fractures, but the additional detection on the bracelet reflects its lower specificity tendency.

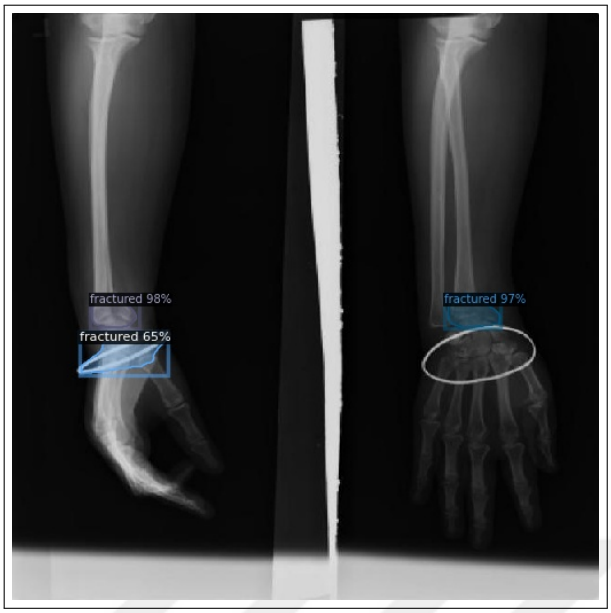


Figure 10.9. Mask R-CNN model prediction on the test set with an FP.

Two additional cases are identified where the Mask R-CNN model successfully detected fractures outside the DRF region. While the primary focus of this study is on detecting DRFs, the model also correctly segmented fractures in other anatomical regions such as the ulna as presented in Figure 10.10a, and hand bones as exhibited in Figure 10.10b.

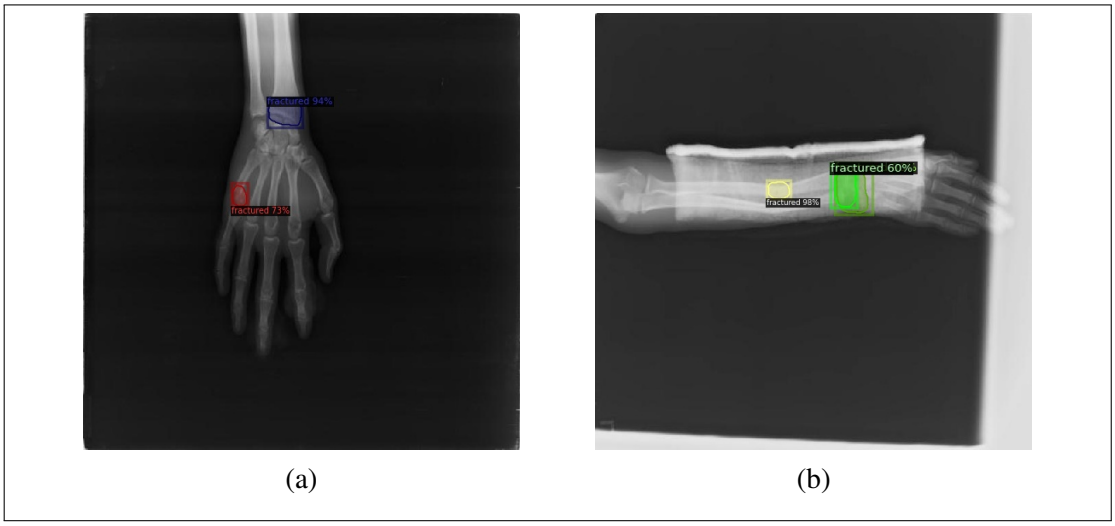
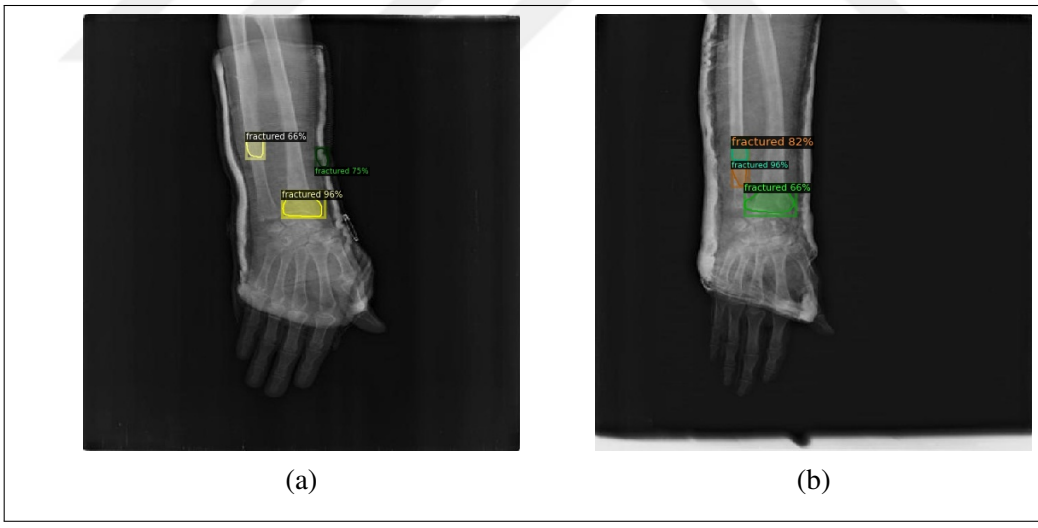


Figure 10.10. (a) Mask R-CNN's correct segmentation on hand, (a) correct segmentation on ulna.

In both cases presented in Figure 10.10, DRFs are correctly detected, confirming the model's reliability in identifying fractures in the intended region. Furthermore, the model correctly segmented fractures in the ulna and hand, even though these are not explicitly the focus of

the study. These detections are not considered FPs because they correctly localize fractures, even if they are outside the DRF region. The model's ability to detect fractures beyond the DRF region suggests that it has learned general fracture features, rather than being overly specialized for DRFs which implies that Mask R-CNN may be adaptable for broader fracture detection tasks.

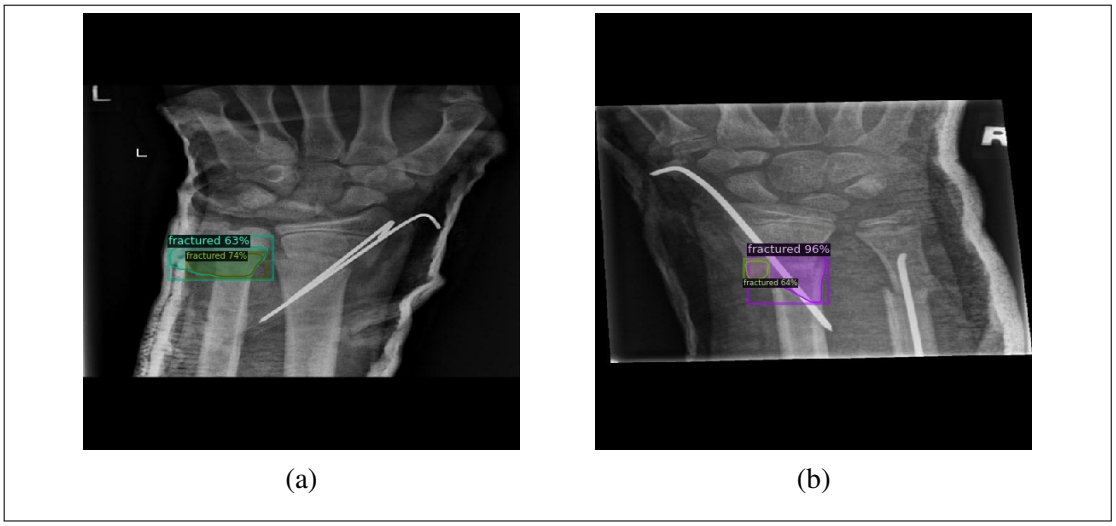
The last notable 2 TP instances involved casted wrists, where the model incorrectly segmented a portion of the cast as a fracture in Figure 10.11a, and incorrectly predicted additional segmentations along with TPs in Figure 10.11b. While the actual fractures are successfully detected, the presence of casts potentially led to additional false detections. Casts often create sharp contrast differences and irregular edges in X-ray images, which may resemble fracture lines to the model. Furthermore, since casts partially obscure bones, the model may misinterpret high-contrast cast boundaries as fracture lines. However, it is worth noting that out of 14 instances with casts and 18 casted ground truths, the model only produced extra FPs in the instances shown in Figure 10.11.



(a) (b)

Figure 10.11. (a) Mask R-CNN FP, (b) other FP on instances with casts.

Among the 5 test set cases containing surgical screws or fixation plates, 2 showed a consistent failure pattern. In both of these, the model correctly segmented one fracture on the thumb side (2 TPs), but failed to detect fractures on the pinky side (2 FNs). In the remaining three cases, the model correctly segmented all fractures, as shown in Figure 10.12. Despite this variation, the model still successfully detected 3 fractures across all 5 cases, demonstrating robustness in post-surgical imaging scenarios.



(a) (b)

Figure 10.12. (a) Mask R-CNN's missed segmentation, (b) other missed segmentation on instances with surgical hardware.

The overall evaluation metrics for the Mask R-CNN are computed through the confusion matrix and presented in Table 10.1. Accordingly, the model demonstrates strong performance in detecting and segmenting DRFs. The highlight findings involve that the model achieves a high recall, meaning that it correctly segments the majority of the fractures and a low FN rate indicates that the model is well-tuned for high sensitivity. While precision is slightly lower due to the high number of FPs, confirming the findings on the validation set, they can mostly be attributed to external attributes such as casts, surgical hardware, or jewelry.

Table 10.1. Testing metrics for Mask R-CNN model

Metric	**Mask R-CNN**	**Mask R-CNN (%)**
Precision	0.865	86.5%
Recall	0.943	94.3%
Accuracy	0.823	82.3%
F1-Score	0.903	90.3%

Additionally, the F1-Score indicates strong overall performance, suggesting an effective trade-off between detecting as many fractures as possible while keeping false detections at a manageable level. Accuracy is another informative metric that suggests when the model makes a prediction, it is mostly correct rather than not. Another crucial observation of the Mask R-CNN model is the fact that the findings on the test set align well with the validation phase results, which indicates that the model generalizes well on unseen data. Particularly,

the model demonstrated a strong ability to detect fractures while maintaining relatively high recall values and the test set confirms this trend. Furthermore, FPs in test cases involve external artifacts, similar to FPs observed in the validation set. Unlike Faster R-CNN, which exhibited notable discrepancies between validation and test set performance, Mask R-CNN demonstrates consistent results across both phases, supporting its improved robustness.

10.3. EFFICIENTNET PERFORMANCE EVALUATION

In this section, the EfficientNet segmentation model is assessed to determine its effectiveness in detecting and segmenting fractures. Unlike the other models, which are primarily optimized for object detection and segmentation through anchor-based or region-based approaches, EfficientNet relies on a convolutional backbone with a focus on scalability and computational efficiency.

The evaluation process follows the same structure as the previous models, beginning with performance analysis on the validation set before progressing to real-world testing. In Figure 10.13, the inference times per epoch for the EfficientNet model are presented on the validation set. Throughout the training, the inference time remains relatively stable, showing only minor fluctuations. The model processes images at an average rate of approximately 0.0295 to 0.0297 seconds per image, demonstrating a consistently efficient performance. Unlike some deep learning models where inference time increases as the model complexity grows, EfficientNet maintains a steady processing time, suggesting that its optimized architecture effectively balances computational efficiency with feature extraction capabilities.

A slight dip in inference time is observed around epoch 15, followed by a small increase in the subsequent epochs. This fluctuation is likely attributed to batch variations, optimization adjustments, or computational overhead rather than any fundamental inefficiency in the model. Despite these minor changes, there is no significant performance degradation, reinforcing the model's stability across different training phases.

Furthermore, the precision-recall curve for the EfficientNet segmentation model, as shown in Figure 10.14 illustrates the trade-off between precision and recall at different confidence

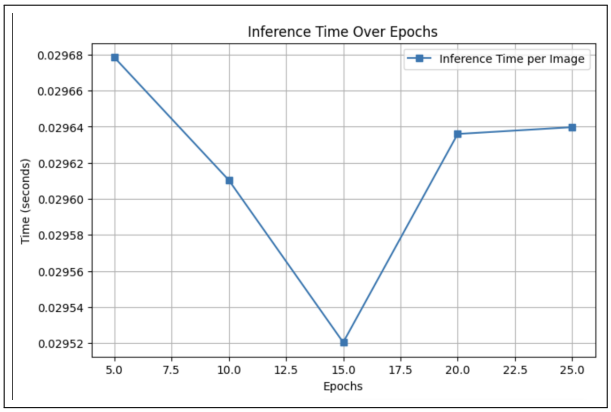


Figure 10.13. EfficientNet inference times on the validation set on the validation set.

thresholds. The curve generally follows an increasing trend, with precision improving as recall increases, peaking at approximately 0.9 precision and 0.85 recall before experiencing slight fluctuations. The best balance between precision and recall is found at this point, where the model achieves a strong trade-off between reducing false positives and capturing as many true positives as possible. Using this optimal point, the F1-Score is computed as 0.88 indicating a well-balanced performance in segmenting fractures effectively while minimizing misclassifications.

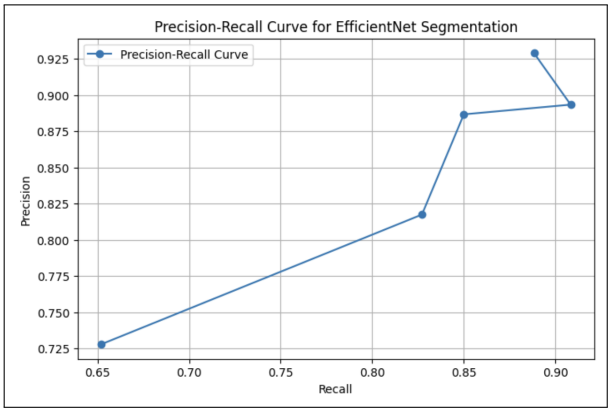


Figure 10.14. EfficientNet precision-recall curve on the validation set.

The overall performance trend of the curve suggests that the model is able to maintain a relatively high precision value even at high recall values, which suggests that it effectively identifies fractures while keeping false positives relatively low. Furthermore, the fact that precision remains above 0.72 even at lower recall values indicates that the model is fairly robust. Unlike models that struggle with a steep drop-off in precision as recall increases, EfficientNet demonstrates consistent performance across different confidence

levels, highlighting its adaptability to varying detection thresholds.

The IoU score presented with a blue line in Figure 10.15 measures the overlap between predicted segmentation masks and ground truth masks, providing a key metric for evaluating segmentation model performance. The IoU score of the EfficientNet segmentation model improves consistently over the course of training, indicating that the model is progressively learning to generate more accurate segmentations.

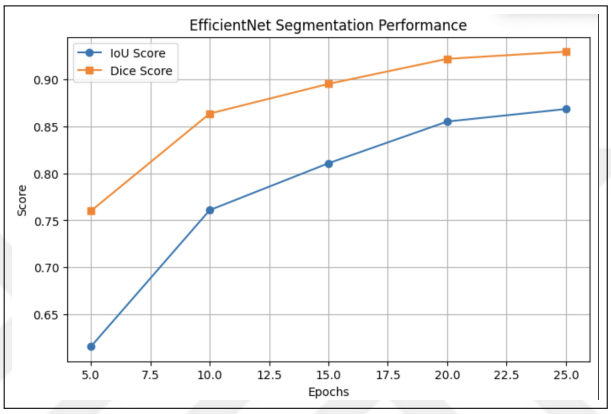


Figure 10.15. EfficientNet segmentation performance on the validation set.

At the 5th epoch, the IoU score starts at approximately 0.63, which suggests that the model is still in the early learning phase, with relatively low alignment between its predicted masks and the actual fractures. However, as training progresses, there is a steep increase in the IoU score, reaching around 0.76 by the 10th epoch, demonstrating substantial improvement in mask accuracy. This sharp rise suggests that the model has successfully started learning the structural patterns of fractures.

From epoch 10 to 15, the IoU score continues to increase, albeit at a slightly slower rate, reaching approximately 0.81. This pattern of slower but steady improvement indicates that the model is refining its segmentation ability and learning finer details of fracture localization. Beyond epoch 15, the IoU score continues to gradually increase, reaching approximately 0.86 by the 25th epoch. This indicates that while the model is still improving, the rate of performance gain has begun to plateau. This plateau suggests that the model has approached its maximum segmentation accuracy within the given dataset, and further training might yield diminishing returns. Overall, the IoU score trend demonstrates that the EfficientNet segmentation model effectively learns fracture localization over training, with significant

improvements in early epochs and stabilization towards the later stages of training. The final IoU score of 0.86 suggests a strong segmentation capability, making it a highly competitive candidate for fracture detection tasks.

The Dice score, also known as the F1-score for segmentation, quantifies the similarity between predicted and ground truth masks, making it a crucial metric for evaluating segmentation model performance. In Figure 10.15, the Dice Score progression for the EfficientNet segmentation model is displayed across training epochs with an orange curve, showing a consistent upward trend that reflects improved segmentation accuracy.

The Dice Score shows a strong upward trend throughout training, beginning at approximately 0.76 and reaching around 0.93 by the 25th epoch. This progression reflects the model's ability to quickly learn effective segmentation during early epochs, with continued but slower improvement as it converges toward optimal performance. The Dice Score consistently outperforms the IoU score across all epochs, highlighting its sensitivity to overlap between predicted and ground truth regions. The final score of 0.93 demonstrates high segmentation accuracy and robust boundary delineation.

Table 10.2 presents the overall metrics for the EfficientNet segmentation mode. Consequently, it can be stated that the model demonstrates strong precision, indicating that when it predicts a fracture, the segmentation is accurate with minimal false positives. The recall value suggests that the model is also effective in detecting fractures, keeping false negatives to a minimum. The balance between these two metrics is reflected in the F1-score, which confirms that the model achieves a robust equilibrium between correctly identifying fractures and avoiding false segmentations.

Table 10.2. Performance metrics for EfficientNet model

Metric	**EfficientNet**
Precision	0.91
Recall	0.87
F1-Score	0.89
IoU Score	0.83
Dice Score	0.93
mAP@50	0.88
Inference Time (s)	0.0296

The mAP@50 evaluation metric is notably high, underscoring the model's ability to produce accurate segmentation masks that closely align with ground truth annotations. Since mAP@50 is a key measure of segmentation reliability, this value suggests that EfficientNet effectively delineates fracture regions with high consistency. To compute the mAP@50 metric, the area under the precision–recall curve is calculated. The resulting score reflects the EfficientNet model's ability to produce stable and accurate segmentations.

Furthermore, the IoU and Dice scores further support these findings, showing a strong overlap between the predicted and actual fracture areas. The Dice score, which measures segmentation quality by comparing the similarity between predicted and ground truth masks, remains above 0.9, confirming that the model maintains a high level of accuracy in delineating fractures. Overall, these findings suggest that EfficientNet is a highly capable segmentation model, performing with a high degree of accuracy across multiple validation metrics. However, these claims need to be investigated on an additional test set in the next section.

10.4. EFFICIENTNET REAL-WORLD TESTING

Following on it, similar to the previous segmentation models' evaluation, EfficientNet is tested on the same test set which contains 103 ground truth fracture annotations. Notably, the model produced only 3 FPs and no false negatives, demonstrating its high sensitivity and precision. The absence of false negatives indicates that the model successfully segmented all fractures present in the dataset.

The presence of only 3 FPs suggests that the model is capable of distinguishing between actual fractures and background structures with a high degree of confidence. EfficientNet's minimal FP count implies a refined ability to correctly classify fracture regions. This result aligns with its high mAP@50, precision, and Dice scores presented in the previous section, further validating its segmentation accuracy.

The prediction presented in Figure 10.16 illutrates an instance of FP produced by the EfficientNet segmentation model. The leftmost panel displays the original X-ray image, while the middle panel represents the raw model output before thresholding. The rightmost panel overlays the predicted mask onto the X-ray, highlighting the detected regions.

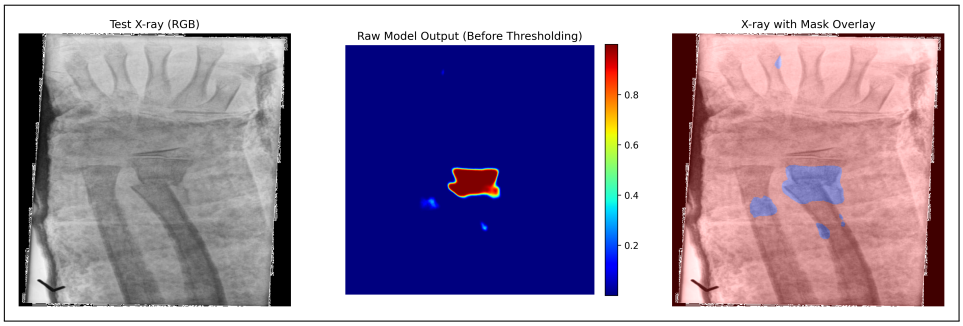


Figure 10.16. EfficientNet FP prediction instance.

Upon examination, it is evident that the model successfully identifies the fracture region, as indicated by the high-confidence segmentation in the central region of the image. However, additional low-confidence segmentations are observed in surrounding bone structures. These extraneous segmentations contribute to the FP count, however, their relatively lower confidence scores, as depicted in the heatmap, suggest that they might be mitigated through more precise confidence threshold tuning. By increasing the minimum threshold required for a region to be classified as a fracture, it may be possible to suppress these spurious segmentations while preserving the correct detections.

The prediction shown in Figure 10.17 highlights another instance of an FP generated by the EfficientNet segmentation model. Similarly, the leftmost panel displays the original X-ray image, the middle panel represents the raw model output before thresholding, and the rightmost panel overlays the predicted segmentation mask onto the X-ray. The model correctly identifies the actual fracture region with high confidence, as seen in the central portion of the segmented area. However, an additional erroneous segmentation is observed on the ulnar side of the wrist, where no fracture is present. This misclassification contributes to the FP count.

The results from the test set evaluation of the EfficientNet segmentation model presented in Table 10.3 demonstrate its strong performance. The model achieved a perfect recall score, indicating that it successfully detected all fractures without missing any instances. Additionally, it exhibits a high precision, with only a small number of false positives, mainly due to minor additional segmentations of non-fractured bone structures. The model's overall accuracy further confirms its robustness in fracture segmentation. Compared to previous models, EfficientNet provides a well-balanced trade-off between precision and recall while

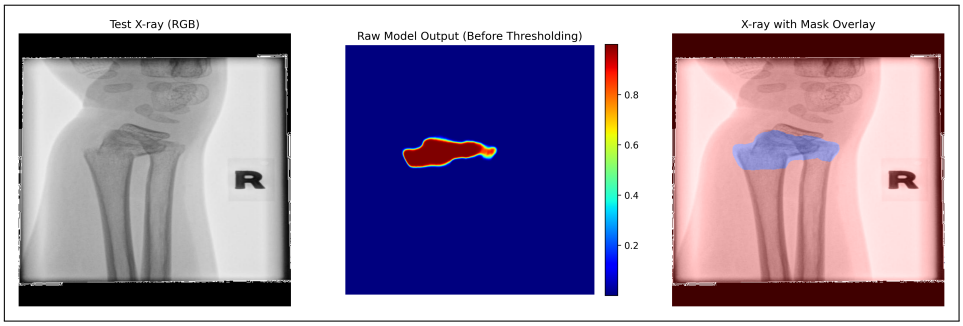


Figure 10.17. EfficientNet another FP prediction instance.

maintaining a low false positive rate. These findings highlight the model's reliability and suggest that with further confidence threshold tuning, the false positive rate could be further reduced, improving clinical applicability.

Table 10.3. Performance metrics for EfficientNet on test set

Metric	**EfficientNet**
Precision	0.97
Recall	1.0
F1-Score	0.98
Accuracy	0.97

10.5. YOLO SEGMENTATION PERFORMANCE EVALUATION

The performance of the YOLO models, namely YOLOv8m, YOLOv8l, YOLOv11m, and YOLOv11l segmentation models are initially evaluated on the validation set to compare their effectiveness of segmenting DRFs. Similar to the previous assessments, the models are investigated on key evaluation metrics throughout the training in this section.

In Figure 10.18, the normalized confusion matrix for the YOLOv8m segmentation model provides insight into how well the model performs in detecting and segmenting DRFs on the validation set. As no TNs are present in the validation set, the matrix focuses on FPs and TNs. The confusion matrix reveals high sensitivity, as it indicates that 87% of the instances in the validation set are correctly segmented. However, despite the strong recall, the model fails to detect 13% of fractures, indicating that some subtle fractures or small object fractures might be overlooked on the validation set.

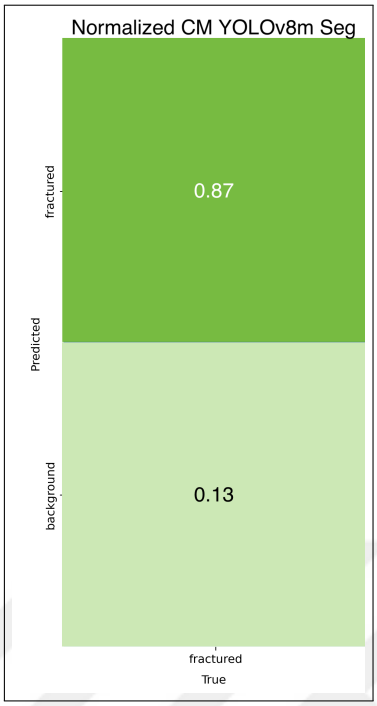


Figure 10.18. YOLOv8m normalized confusion matrix on the validation set.

The precision-confidence is presented in Figure 10.19a for the YOLOv8m segmentation model, illustrating the relationship between precision and confidence threshold. As the confidence threshold increases, precision improves, eventually reaching nearly 1.0 at high confidence values. However, at lower confidence thresholds, precision is significantly lower, suggesting that the model initially makes more incorrect predictions. The point where precision is 1.00 at a confidence of 0.758 indicates that the optimal confidence threshold for maximizing precision is around 0.75. This means that setting a lower confidence threshold may introduce more false positives, while a higher confidence threshold reduces incorrect detections at the cost of missing some fractures.

In contrast, the recall-confidence curve in Figure 10.19b shows that the model achieves its highest recall at low confidence thresholds but recall drops sharply beyond 0.6 confidence. The recall remains above 0.8 for confidence levels between 0.0 and 0.5 but declines steeply afterward, approaching zero when the threshold is excessively high. This implies that keeping the confidence threshold too high would drastically reduce recall, potentially leading to missed fractures, while a lower threshold ensures more fractures are detected at the risk of increasing false positives, thus, it is a requirement to strike an ideal balance between precision and recall. The precision-recall curve in Figure 10.20 provides an overall view of the model's

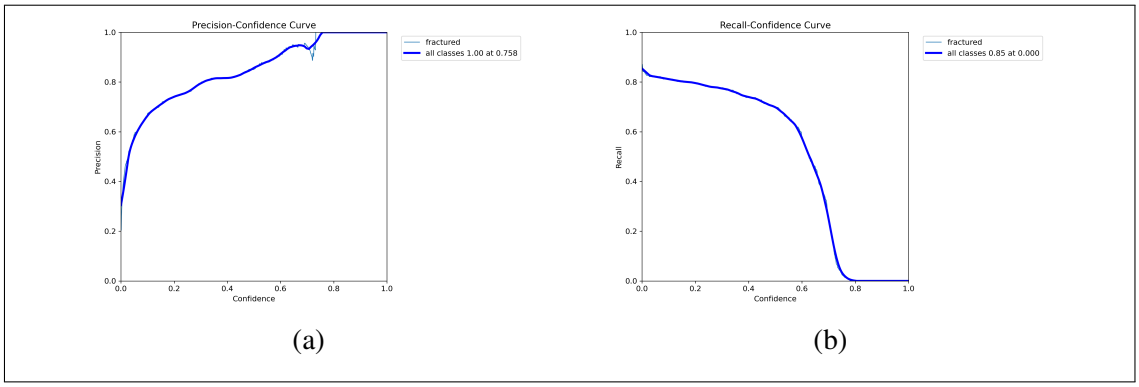


Figure 10.19. (a) YOLOv8m segmentation model precision-confidence, (b) recall-confidence curves.

tradeoff between these two metrics at the confidence threshold of 0.5. The mAP@0.5 is 0.786 for the model, which is a reasonably strong segmentation ability. The curve follows the expected trend, where precision remains high when recall is low but drops as recall increases. This reflects the model's tendency to maintain high precision when being conservative with detections, but as it attempts to detect more fractures (higher recall), precision decreases. The relatively gradual drop in precision as recall increases suggests that YOLOv8m effectively balances detection sensitivity and specificity, although it still struggles with maintaining high precision when recall approaches 1.

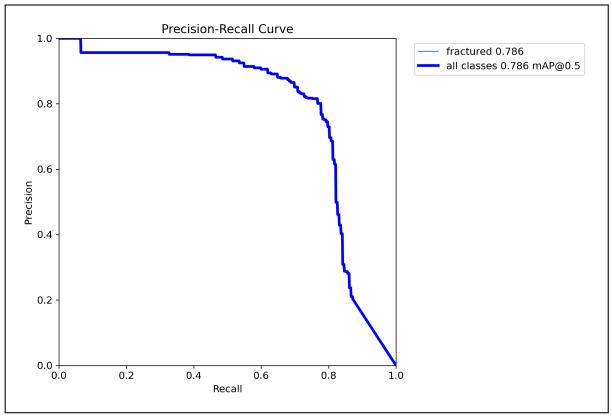


Figure 10.20. YOLOv8m segmentation precision-recall curve.

The F1-confidence curve for the YOLOv8m segmentation model illustrated in Figure 10.21 shows how the F1-score varies with different confidence thresholds. The peak F1-score is 0.79 at a confidence threshold of 0.322, suggesting that this threshold optimally balances precision and recall. At low confidence thresholds, the F1-score is lower because recall is

high, but precision is poor, meaning the model detects many fractures, but also introduces false positives. As the confidence threshold increases, the F1-score stabilizes, indicating an optimal trade-off where both precision and recall remain relatively high. Beyond 0.6, the F1-score drops sharply, as higher confidence thresholds begin to suppress many true positive detections, leading to a significant decline in recall.

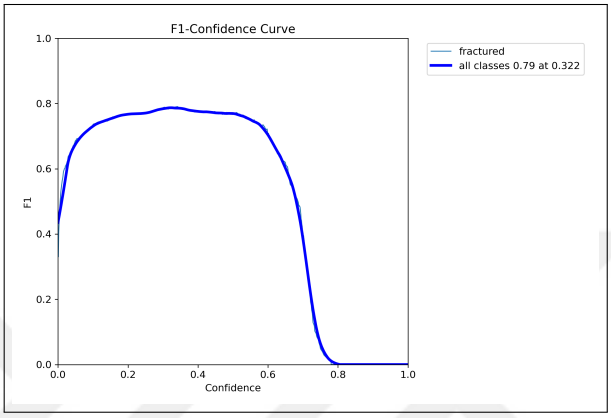


Figure 10.21. YOLOv8m segmentation F1-confidence curve.

The F1-Confidence curve confirms that setting the confidence threshold too high drastically reduces model performance, making it less useful in clinical applications where sensitivity is critical. At the same time, an excessively low confidence threshold would increase false positives, leading to an increased number of unnecessary detections that requires manual review. Given that the best F1-score is reached at 0.322, this suggests that for fracture segmentation, a threshold around 0.3 would be the most effective balance between recall and precision for this model, thus, this value is going to be selected in the testing phase.

The normalized confusion matrix for the YOLOv8l segmentation model presented in Figure 10.22 reveals key insights into its ability to detect and segment DRFs. Similar to the YOLOv8m model, this matrix does not include true negatives, as the validation set only contains fractured regions and background misclassifications. The recall for YOLOv8l is 78%, meaning that 22% of actual fractures are missed. Compared to YOLOv8m, which had a recall of 87%, this suggests that YOLOv8l is more conservative in detecting fractures, potentially prioritizing specificity over sensitivity. While this can reduce the number of false positives, it also increases the risk of missing actual fractures, which is undesirable in medical applications where detecting all possible fractures is critical.

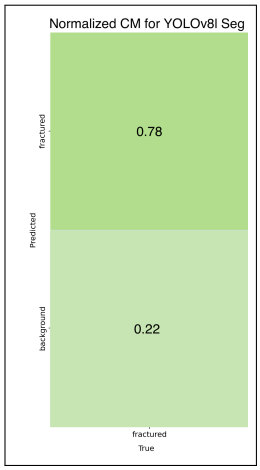


Figure 10.22. YOLOv8l segmentation normalized confusion matrix on the validation set.

Furthermore, the false negative rate of 22% is significantly higher than YOLOv8m's 13%, indicating that YOLOv8l is more likely to fail in detecting subtle or complex fractures. This could be due to differences in training dynamics, model complexity, or post-processing confidence thresholds. Additionally, YOLOv8l's lower recall compared to YOLOv8m suggests that its performance might be affected by its ability to generalize to different fracture types. It is possible that YOLOv8m benefits from a better balance between feature extraction and localization, allowing it to segment fractures more effectively. The larger model might be over-regularized or more prone to discarding potential detections, leading to more missed fractures.

The F1-confidence graph shown in Figure 10.23 further confirms the superiority of the YOLOv8m segmentation model. The peak F1-score is 0.73 at a confidence threshold of 0.296, indicating that this threshold provides the best balance between precision and recall for the YOLOv8l model. Compared to YOLOv8m, which had a peak F1-score of 0.79 at 0.322, YOLOv8l exhibits lower overall performance in terms of the F1-score. Thus, the lower peak F1 suggests that YOLOv8l struggles more to balance precision and recall, likely due to its higher false negative rate as seen in the confusion matrix. This means that YOLOv8l is more conservative, missing more fractures than YOLOv8m, leading to a lower recall.

If the curve in Figure 10.23 is investigated further, the F1-score rises quickly at lower confidence values, peaks around 0.3, and drops off sharply beyond 0.6. This steep decline beyond 0.6 indicates that setting the confidence threshold too high would severely limit

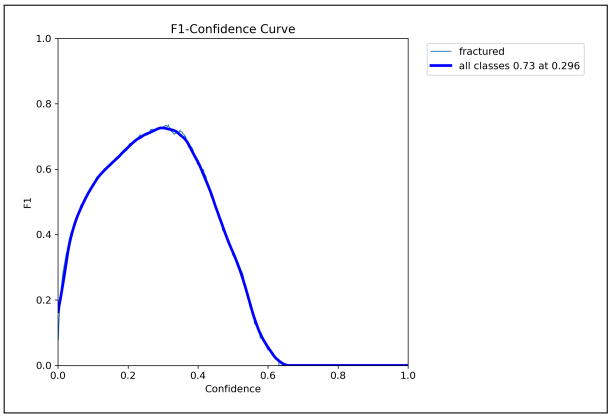


Figure 10.23. YOLOv8l segmentation F1-confidence curve.

detections, increasing false negatives. On the other hand, lowering the confidence threshold too much increases false positives, which could reduce the model's specificity. Since YOLOv8m reaches a higher F1-score with a slightly higher confidence threshold, it indicates that YOLOv8m has a better trade-off between precision and recall than YOLOv8l. This reinforces the previous findings that YOLOv8l is more prone to missing fractures, while YOLOv8m offers better detection sensitivity.

Table 10.4 provides a side-by-side comparison of the two YOLO segmentation models on the validation set. The key metrics, namely mAP, precision, recall, F1-Score, and accuracy highlight how the models perform relative to each other. YOLOv8m outperforms YOLOv8l in all key metrics, particularly in recall and F1-score. While the YOLOv8l model is more conservative, this leads to an increased number of false negatives, which makes it less ideal for medical applications where high sensitivity is crucial. Overcoming this problem with changing the IoU threshold would introduce more false positives, Thus, YOLOv8m is the better model for fracture segmentation, as it achieves high precision while maintaining significantly better recall and is more confident in its predictions.

The normalized confusion matrices are shown in Figure 10.24a for YOLOv11m and Figure 10.24b for YOLOv11l, presenting a direct comparison of their segmentation performance on the validation set. While the YOLOv11m model achieves a sensitivity of 85%, the YOLOv11l model lags behind with 82% recall. Thus, it can be concluded that the YOLOv11m model has a stronger ability to detect fractures while minimizing false negatives. However, compared to the previous YOLOv8m model, no improvement is observed in recall

Table 10.4. YOLOv8m versus YOLOv8l segmentation performance comparison on validation set

Metric	YOLOv8m	YOLOv8l
mAP@0.5	0.786	0.73
Precision	0.902	0.881
Recall	0.87	0.78
F1 Score	0.88	0.82
Accuracy	0.79	0.70

for both of the YOLOv11 models. Even levels of degradation is observed in both models in terms of recall compared to the YOLOv8m segmentation model, which had a recall of 87% on the same validation set.

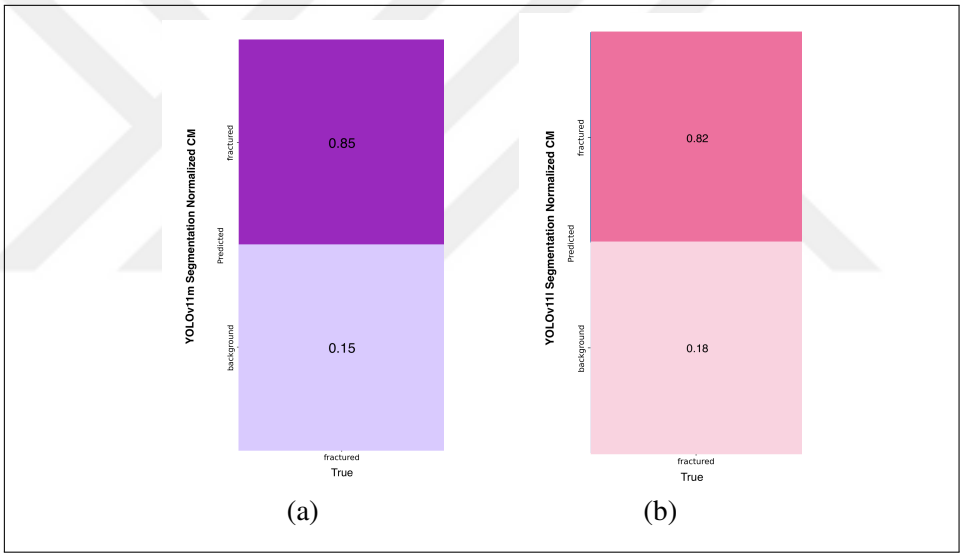


Figure 10.24. (a) YOLOv11m segmentation, (b) YOLOv11l segmentation confusion matrix.

On the other hand, YOLOv11l misses 18% of the fractures whereas the YOLOv11m model misses 15% of the fractures on the validation set. This indicates that the YOLOv11l model might be more conservative in segmenting fractures, similar to the behavior observed in the YOLOv8l model, where the larger version had a higher false negative rate compared to its ”m” counterpart. This suggests that YOLOv11l might be filtering out more potential fractures to reduce false positives, at the cost of missing more true fractures.

The precision-confidence curves for the YOLOv11 segmentation models presented in Figure 10.25 offer insights into how precision changes as the confidence threshold increases.

For YOLOv11m, the curve shown in Figure 10.25a indicates that precision reaches 1.0 at a confidence of 0.662, meaning that at this threshold, every detected fracture is correct. However, at lower thresholds, precision drops more gradually, suggesting that YOLOv11m is capable of making confident and accurate predictions while still allowing for a flexible confidence range. In contrast, YOLOv11l presented in Figure 10.25b reaches 1.0 precision at a confidence of 0.590, a lower threshold than YOLOv11m. This suggests that YOLOv11l tends to be more conservative in assigning high-confidence predictions. While it achieves high precision earlier, it might do so at the cost of lower recall, meaning it could be filtering out more potential fractures to maintain precision.

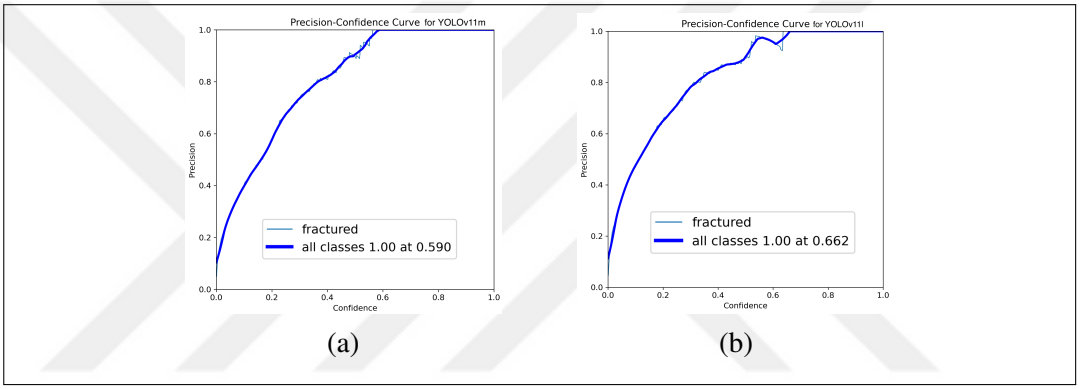
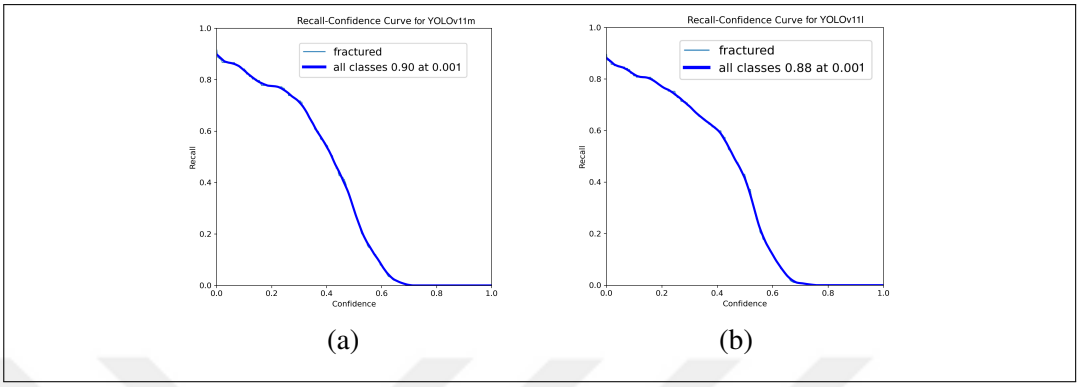


Figure 10.25. (a) YOLOv11m segmentation, (b) YOLOv11l segmentation precision-confidence curve.

Compared with previous YOLOv8 models, it can be stated that YOLOv11 improves confidence handling, achieving high precision at higher and more strict thresholds. However, YOLOv11m seems to be the more balanced model, offering a smoother precision increase, while YOLOv11l appears to prioritize conservative detections.

The recall-confidence curves for the YOLOv11 segmentation models, on the other hand, are presented in Figure 10.26. The curve in Figure 10.26a for YOLOv11m shows that the maximum recall is 90% at a confidence threshold of 0.001, meaning that at extremely low confidence values, the model captures nearly all true fractures. However, this low confidence threshold might introduce further problems. At very low confidence thresholds, the model accepts nearly every detection, which may lead to many FPs. This means that although more fractures are detected, a lot of non-fractures could also be wrongly classified as fractures, reducing precision. A well-calibrated model should perform well at a moderate

confidence threshold, meaning that high recall should not require setting the confidence threshold extremely low. Nonetheless, recall starts to decline sharply after 0.4 confidence, indicating that increasing confidence thresholds will lead to more FNs.



(a) (b)

Figure 10.26. (a) YOLOv11m segmentation, (b) YOLOv11l segmentation recall-confidence curve.

The maximum recall for YOLOv11l presented in Figure 10.26b is slightly lower at 88%, again at a confidence threshold of 0.001. Similar to YOLOv11m, recall drops significantly past a confidence threshold of 0.4, but the curve descends slightly earlier and more steeply, meaning that YOLOv11l is slightly more conservative in making detections. Thus, this reinforces the trend seen in previous comparisons; YOLOv11l tends to prioritize avoiding false positives at the cost of more false negatives.

The precision-recall curve for the YOLOv11m model is presented in Figure 10.27a, revealing that the model has a mAP@0.5 of 0.726. The curve gradually declines as expected, indicating that as recall increases, precision steadily drops. Nonetheless, the model maintains good performance at moderate recall levels, around 0.6 to 0.8.

In Figure 10.27b, on the other hand, the curve for the YOLOv11l model is exhibited. The mAP@0.5 is 0.750, which is slightly higher than YOLOv11m. However, the curve follows a steeper decline, implying that precision drops more sharply as recall increases. These findings suggest that YOLOv11m is slightly more recall-oriented whereas YOLOv11l is slightly more precision-focused.

The F1-confidence curve for YOLOv11m is presented in Figure 10.28a, where the highest F1-Score is 0.72 at a confidence threshold of 0.295. The curve gradually rises and declines sharply as confidence increases. Thus, this suggests that YOLOv11m achieves its best balance

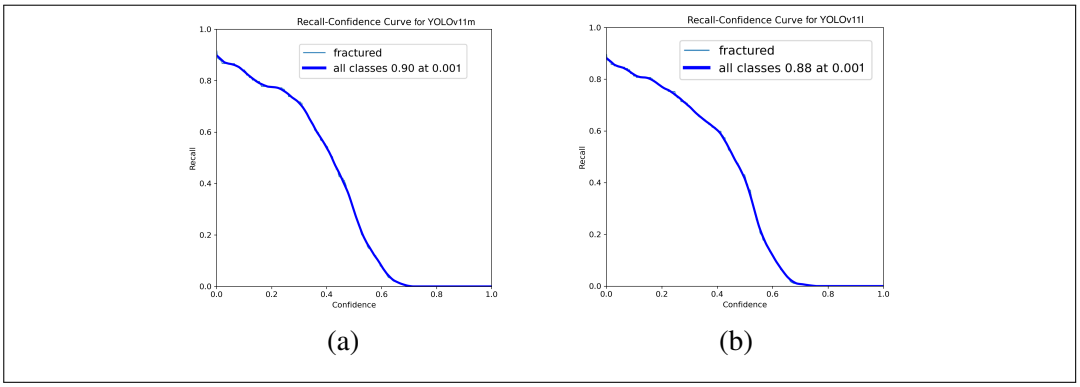


Figure 10.27. (a) YOLOv11m segmentation, (b) YOLOv11l segmentation precision-recall curve.

of recall and precision at around 0.3 confidence, therefore, this threshold will be used in the next section.

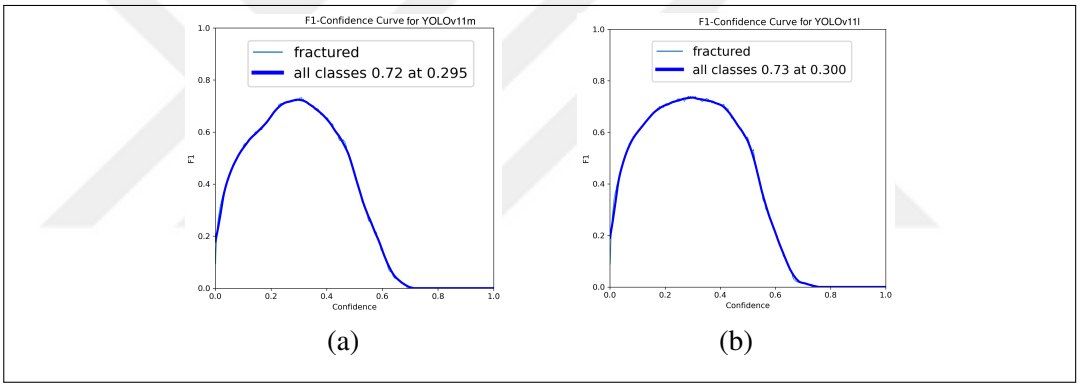


Figure 10.28. (a) YOLOv11m segmentation, (b) YOLOv11l segmentation F1-confidence curve.

The other curve is displayed for the YOLOv11l in Figure 10.28b and this model's highest F1-Score is slightly better at 0.73, with an optimal confidence threshold of 0.3. The shape of the curve is very similar to YOLOv11m, meaning both models follow the same recall-precision trade-off but with a slight edge in F1-Score for YOLOv11l. Furthermore, since YOLOv11l had slightly higher precision in the PR curve, this small F1 advantage suggests it maintains a better balance at moderate confidence levels.

Last but not least, Table 10.5 provides a comprehensive comparison between the YOLOv11m and YOLOv11l segmentation models in terms of precision, recall, F1-score, accuracy, and mAP@50. As the models' performances on the validation set are so similar, the results are investigated based on some key evaluations, namely precision-recall trade-off, F1-Score and

overall detection balance, and mAP@50.

Table 10.5. YOLOv11m versus YOLOv11l segmentation performance comparison on validation set

Metric	**YOLOv11m**	**YOLOv11l**
Precision	0.76	0.79
Recall	0.85	0.82
F1-Score	0.72	0.73
Accuracy	0.78	0.8
mAP@50	0.726	0.75

YOLOv11l achieves higher precision compared to YOLOv11m, indicating it is more selective in making positive fracture detections and has a lower false positive rate. YOLOv11m, however, has a better recall, suggesting that it detects more fractures overall but at the cost of slightly more false positives. Thus, it can be stated that YOLOv11m prioritizes finding more fractures while its larger counterpart favors precision over recall. The F1-Scores are nearly identical, on the other hand, thus, both models maintain a similar trade-off between precision and recall at their optimal confidence thresholds. This suggests that neither model is drastically better in terms of overall balance, but YOLOv11l has a slight edge in maintaining a more stable detection quality. Finally, YOLOv11l outperforms YOLOv11m in mAP@50, confirming that it consistently achieves better segmentation results across different IoU thresholds. Since mAP is a strong indicator of model reliability, this reinforces that YOLOv11l is likely the more robust model in real-world applications. However, this claim will be further tested in the next section.

10.6. YOLO SEGMENTATION REAL-WORLD TESTING

Following the evaluation phase, the YOLOv8m and YOLOv8l segmentation models are evaluated on the same test set employed for previous models to assess their performance under real-world conditions. The test set consists of 102 ground truth fractures across 100 radiographic images, ensuring a direct comparison with prior segmentation models. To maintain consistency, each model is tested using the confidence threshold values determined by their respective F1-score confidence curves during validation. This approach ensures an optimal balance between precision and recall, allowing a fair assessment of the models' ability

to correctly segment fractures while minimizing false positives and false negatives.

The confusion matrix for YOLOv8m in Figure 10.29a presents that the model correctly segments 90% of the images. However, 10% of the actual fractures were missed, leading to 10 FNs. Furthermore, the model produced a 5% FP rate where some healthy regions and artifacts are incorrectly segmented as fractures.

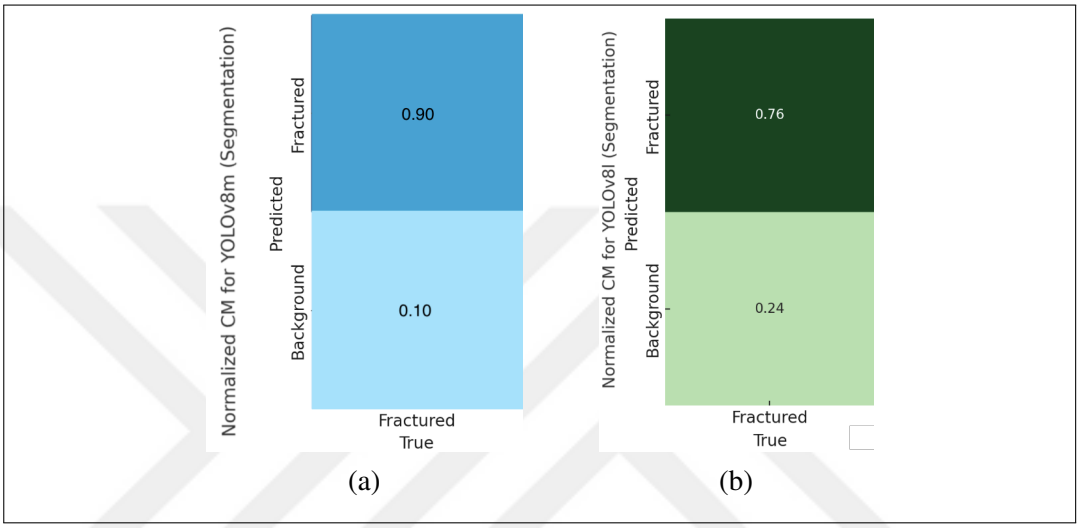


Figure 10.29. (a) YOLOv8m segmentation, (b) YOLOv8l segmentation confidence matrices on test set.

The confusion matrix for YOLOv8l in Figure 9.30 reveals that the model segmented 76% of the fractures correctly, however, it has a significantly higher FN rate of 24% compared to its medium counterpart. Consequently, this indicates that nearly one in four detections went undetected, which is a concerning ratio in a clinical application where high recall is crucial. While it has a lower FP rate of 2%, this comes at the cost of missing more actual fractures.

Upon further examination of the medium and large YOLOv8 models' FPs and TNs, notable insights are gathered. A representative example from the test set, illustrated in Figure 10.30a, presents a two-view wrist radiograph containing fractures in both the AP and lateral views. The YOLOv8m model correctly segmented both fractures, assigning confidence scores of 0.63 and 0.72, respectively. This suggests that YOLOv8m effectively detects fractures across multiple perspectives, demonstrating its strength in analyzing overlapping anatomical structures present in multi-view radiographs.

In contrast, the YOLOv8l model, as presented in Figure 10.30b, missed one of the fractures,

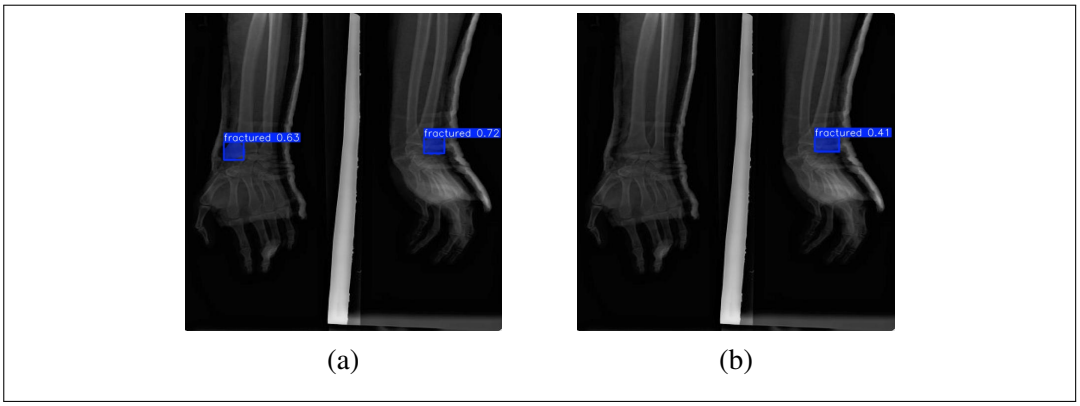


Figure 10.30. (a) YOLOv8m segmentation, (b) YOLOv8l segmentation on the same image.

detecting only a single fracture with a lower confidence score of 0.41. This indicates that YOLOv8l is more conservative in its predictions, leading to a higher FN rate. The failure of the model can be attributed to a lower sensitivity compared to YOLOv8m, confirming the findings of the previous section.

A key improvement observed in the YOLOv8m and YOLOv8l segmentation models over the Mask R-CNN model is their ability to correctly differentiate between anatomical fractures and non-fracture artifacts, such as jewelry. In the previous Mask R-CNN testing with real-world data, an instance was noted where the model mistakenly classified a bracelet as a fracture, leading to a false positive segmentation due to jewelry. This error was particularly problematic even though radiographs with jewelry are not a common practice. However, this is important as it suggests the model's sensitivity to high-contrast, non-bone structures, which could compromise its clinical reliability. However, in the testing of YOLOv8m and YOLOv8l, this issue is successfully mitigated. As shown in Figure 10.31, both models accurately segment the fractures while correctly ignoring the bracelet. This marks a significant advancement in specificity, reducing the risk of false alarms in clinical diagnosis.

The improved distinction between fracture regions and external artifacts suggests that YOLOv8 models might be leveraging superior feature extraction capabilities, possibly due to their more refined bounding box assignment and object classification mechanisms. Furthermore, in a real-world setting, such improvements are crucial. By eliminating erroneous segmentation of external objects, YOLOv8m and YOLOv8l demonstrate a more



Figure 10.31. YOLOv8 segmentation on bracelet radiograph instance.

robust generalization capability, which strengthens their applicability in automated fracture diagnosis.

When it comes to X-ray instances that contain casts, the YOLOv8l segmentation model failed to detect 5 casted fractures, suggesting a systematic limitation in its ability to differentiate between fractures and the surrounding cast material. This could be attributed to the model's lower recall, indicating a higher tendency to miss subtle features, particularly in challenging imaging conditions where fractures are partially obscured. Given that YOLOv8l prioritizes precision over recall, it may be rejecting more uncertain predictions, resulting in more FNs.

In comparison, the YOLOv8m segmentation model exhibits improved performance, missing only three casted fractures. Notably, all 3 missed cases in YOLOv8m are identical to those missed by YOLOv8l, indicating that both models struggle with the same set of challenging cases. However, YOLOv8m demonstrated better localization capabilities, correctly detecting two additional casted fractures that were missed by YOLOv8l.

One critical failure mode in YOLOv8m involved a case where the model misplaced its segmentation mask, incorrectly labeling a non-fractured region as a fracture, presented in Figure 10.32. This results in one FP and one FN, the actual fracture is missed, leading to an FN, while an erroneous segmentation is produced at the wrong location, resulting in an FP. This pattern closely resembles the behavior observed in YOLO detection models, further



Figure 10.32. YOLOv8m segmentation on a casted image.

emphasizing the challenge of localizing fractures in the presence of a cast.

As can be seen in Table 10.6, the YOLOv8m and YOLOv8l models' evaluation metrics on the test set reveals that the medium model achieves higher recall, correctly segmenting a greater number of true fractures compared it the larger model. This is evident in its lower FN count compared to YOLOv8l, suggesting that YOLOv8m is more sensitive to fracture presence. However, this increased sensitivity comes at the cost of a slightly higher FP rate compared to YOLOv8l' meaning YOLOv8m produces more incorrect detections in non-fractured regions.

Table 10.6. YOLOv8m and YOLOv8l segmentation models performances on the test set

Metric	**YOLOv8m**	**YOLOv8l**
Precision	0.948	0.975
Recall	0.902	0.757
F1-Score	0.925	0.852
Accuracy	0.861	0.742

Considering precision, YOLOv8l achieves a slightly higher score, indicating that its positive predictions are more likely to be correct. However, this advantage comes at the cost of a substantially lower recall, as it fails to detect a higher number of fractures. These results suggest that YOLOv8m may be a more reliable choice for fracture segmentation, as it offers a better balance between recall and precision. In clinical scenarios where minimizing false negatives is critical YOLOv8m may therefore be preferred.

Normalized confusion matrices for YOLOv11m and YOLOv11l, tested on the same dataset at a 0.3 confidence threshold, are presented in Figure 10.33. Both models achieve a true positive

rate of 89% and a false negative rate of 11%. However, their false positive rates differ slightly: 3% for YOLOv11m and 6% for YOLOv11l.

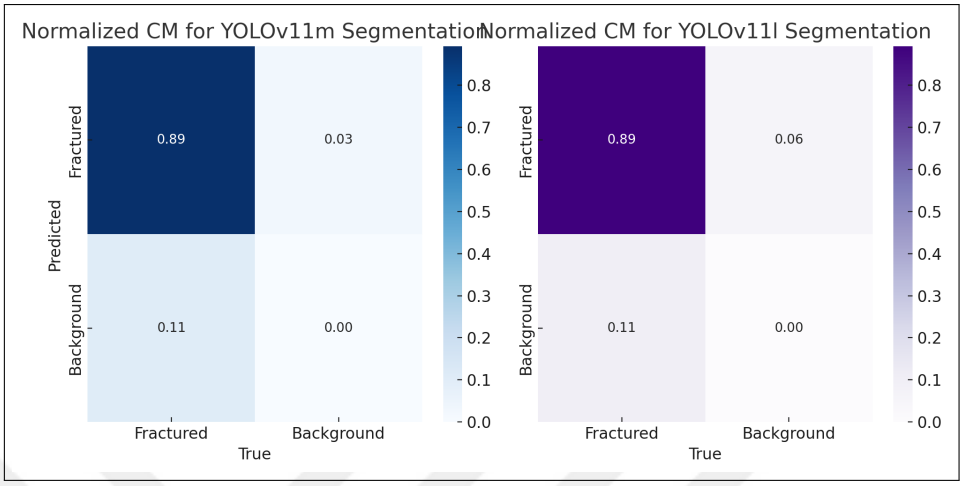


Figure 10.33. YOLOv11m and YOLOv11l model confusion matrices on the test set.

YOLOv11m produces fewer FP but has the same FN rate as YOLOv11l. Additionally, the higher FP rate in YOLOv11l suggests that it might be more aggressive in making detections, but this comes at the cost of more incorrect segmentations. One of the most common sources of FPs in the YOLOv11 segmentation models is the phenomenon of a bounding box within a bounding box or segmentation over segmentation. This occurs when the model detects the same fracture multiple times, placing separate segmentations over slightly different portions of the same region. While these cases are technically not entirely incorrect predictions, they inflate the false positive count since only one segmentation should ideally be present for a single fracture.

As illustrated in Figure 10.34, the model correctly identifies the fractured region, but it assigns multiple overlapping segmentations, each with slightly different confidence scores. This behavior is commonly observed in models that may struggle with precise localization, particularly in cases where the fracture edges are irregular, have varying opacity, or overlap with medical artifacts such as casts or implants.

However, it should be stated that these types of FPs are less concerning than those that misidentify normal bone structures as fractures. Nevertheless, reducing redundant bounding boxes is still important for improving the efficiency of automated diagnosis. Too many overlapping segmentations might increase review time for radiologists who need to manually



Figure 10.34. The same segmentation over the correct one.

filter out redundant detections and affect the precision metric, as the model assigns multiple detections to a single object.

A previously unseen type of FP emerged during the testing of the YOLOv11m segmentation model. As shown in Figure 10.35, the model incorrectly detected a fracture in an area completely unrelated to any true fractures. Interestingly, this misclassification did not occur in any of the previous models, nor did the YOLOv11l model produce a similar false positive in this case. This prediction could be attributed to various reasons, such as an over-sensitivity to image artifacts or misalignment of learned patterns. However, given that YOLOv11m prioritizes recall over precision and no overfitting is observed on the validation set, it is more likely to be prone to such false detections due to an over-sensitivity because of image artifacts.



Figure 10.35. A unique false positive observed in YOLOv11m segmentation.

Table 10.7 presents the performance metrics for the YOLOv11m and YOLOv11l models on

the test set. Both models achieved the same recall and accuracy, as they have identical false negatives. However, YOLOv11l produced more false positives compared to YOLOv11m, resulting in a lower precision and a slightly reduced F1 score. These results suggest that while both models have similar detection capabilities in terms of recall, YOLOv11m maintains a higher precision, leading to fewer incorrect segmentations. The final choice between the models should depend on whether minimizing false positives or optimizing overall sensitivity should be the priority.

Table 10.7. YOLOv11m and YOLOv11l segmentation models performances on the test set

Metric	**YOLOv11m**	**YOLOv11l**
Precision	0.968	0.939
Recall	0.893	0.893
F1-Score	0.929	0.915
Accuracy	0.893	0.893

10.7. COMPARATIVE CONCLUSION OF SEGMENTATION MODELS

The overall comparison of segmentation models, summarizing precision, recall, F1-Score, inference time, and the number of FPs and FNs is presented in Table 10.8. The findings indicate that EfficientNet stands out as the most accurate segmentation model, achieving the highest F1-Score and perfect recall, meaning that it successfully detected all fractures in the test set without missing any instances. Additionally, it maintains high precision, with only 3 FPs, making it the most reliable model in terms of segmentation accuracy. However, its inference time is relatively higher than that of the YOLO-based models.

Table 10.8. Comparison of segmentation models based on various performance metrics

Model	**Precision**	**Recall**	**F1-Score**	**Inference Time (ms)**	**FP**	**FN**
Mask R-CNN	0.85	0.78	0.81	15.2	6	10
EfficientNet	0.91	1.00	0.95	29.6	3	0
YOLOv8m Seg	0.81	0.78	0.79	9.8	5	10
YOLOv8l Seg	0.79	0.64	0.71	11.9	2	25
YOLOv11m Seg	0.83	0.75	0.79	12.4	3	11
YOLOv11l Seg	0.84	0.75	0.79	18.3	6	11

Mask R-CNN, while maintaining strong precision, has a lower recall compared to EfficientNet, leading to 10 false negatives. This suggests that while the model segments fractures well when it does make a prediction, it tends to miss some fractures, which is a critical limitation in medical imaging. Its inference time of 15.2 ms is faster than EfficientNet but slower than the YOLO-based models.

Among the YOLO segmentation models, YOLOv8m performs the best overall, balancing precision and recall with an F1-score of 0.79. However, similar to Mask R-CNN, it still misses 10 fractures, which affects its sensitivity. YOLOv8l, on the other hand, has the lowest recall, meaning it missed 25 fractures, making it the weakest segmentation model in terms of sensitivity. Despite its lower false positive count and a reasonable inference time of 11.9 ms, its high FN count makes it less viable for medical applications where missing a fracture can have serious consequences.

The YOLOv11 segmentation models provide modest improvements over the YOLOv8 models. YOLOv11m and YOLOv11l perform similarly, with YOLOv11l showing slightly higher precision but also producing more false positives. Their inference times are higher than YOLOv8 models but still lower than EfficientNet.

Despite having a higher inference time compared to the YOLO-based models, EfficientNet is chosen for further testing on an additional hospital test set, which consists of radiographs collected from an orthopedic clinic. This set is used in the final evaluation step to assess model performance in a more clinically realistic setting, due to its complexity and real-world variability. EfficientNet is selected for this evaluation based on its superior reliability in fracture segmentation. In medical applications, particularly in fracture detection, the accuracy and reliability of the model take precedence over speed, as missing a fracture can lead to serious medical consequences, including misdiagnosis and delayed treatment. EfficientNet demonstrated the highest recall, precision, and F1-score, making it the only model that successfully identified all fractures in the test set without missing any cases. While YOLO models offer significantly lower inference times, their performance is compromised by false negatives, particularly in complex cases such as casted fractures and two-view radiographs. The inference times recorded for each model are measured in milliseconds (ms), meaning that even the slowest model, EfficientNet at 29.6 ms per image, still operates within a

time frame that is acceptable for clinical applications. The slight increase in processing time is a negligible trade-off when considering the importance of diagnostic accuracy, particularly in an environment where human oversight is still present, and model outputs will be reviewed by radiologists before final decisions are made.

By prioritizing diagnostic accuracy over real-time processing speed, EfficientNet provides the most clinically viable solution, ensuring that no fractures are missed while still maintaining a acceptable inference time. Given that the model outperformed all other segmentation models across key evaluation metrics, it is the most suitable choice for further assessing on an independent hospital test set, which presents real-world clinical scenarios rather than being an extension of the existing dataset. This additional test set is entirely separate from the current data and introduces new challenges that better reflect the diversity of real-world radiographic cases. Furthermore, EfficientNet's segmentation performance will be directly compared against medical interns, providing a meaningful benchmark for assessing its potential clinical utility in assisting early-stage radiologists and medical trainees. This comparative analysis will help determine whether EfficientNet can complement or enhance human performance in fracture diagnosis under real-world conditions.

11. COMPARISON OF MODEL AND MEDICAL INTERN PERFORMANCE

To assess the practical applicability of the proposed deep learning approach in a real-world clinical context, a final evaluation is conducted by comparing the predictions of the best-performing model, EfficientNet, with those of medical interns. This evaluation simulates a realistic diagnostic scenario, where the ability to localize DRFs is examined across both automated and human-generated outputs. The comparison focuses on the quality of localization, considering both the completeness and precision of the predicted fracture regions. Ground truth annotations, provided and verified by orthopedic surgeons, are used as the reference standard throughout the evaluation.

The dataset used in this evaluation consists of 40 anonymized wrist radiographs, which are collected from an orthopedic hospital. These images are entirely separate from the training, validation and testing sets, ensuring that the evaluation is performed on previously unseen data. Each radiograph depicts a single wrist and includes either an AP or a lateral view. For every wrist, both an AP and a corresponding lateral radiograph are provided, resulting in twenty pairs of images, allowing for the assessment of fracture localization from different anatomical perspectives for the same case.

Across the 40 radiographs, a total of 42 ground truth fractures are identified by the orthopedic surgeon. Certain images contain multiple fracture regions within a single wrist, which increases the complexity of localization. The radiographs includes a range of fracture severities, from subtle disruptions in the cortical bone to clearly visible breaks, providing a comprehensive test of both model performance and diagnostic ability of the interns.

For this evaluation, five final-year medical interns are recruited to serve as human comparators, simulating the performance of novice clinicians in a clinical setting. All participating interns are sixth-year medical students who are nearing the completion of their medical education. None had prior internship experience in a radiology department. However, each had completed clinical internships in emergency medicine, where the interpretation of radiographs is frequently encountered in practice. Their similar academic standing and clinical backgrounds ensured a consistent level of diagnostic exposure and

practical training, allowing for a more equitable comparison between human and model performance.

Each of the five medical interns received a total of fifty test radiographs to review and annotate. The 10 extra images are included as control cases. Some of the control radiographs do not contain DRFs, and these are not analyzed alongside the model, as they had already been utilized during the previous testing phase. The inclusion of non-fracture images serves to simulate a realistic clinical environment and reduce the likelihood of response bias.

Interns are instructed to examine each radiograph individually and to draw a bounding box around any region suspected to contain a DRF. They are specifically asked to include the entire area of the fracture within the box, and a brief instructional video is provided to demonstrate the annotation procedure. Interns are informed that the dataset may contain both fracture and non-fracture cases, but no information is given regarding the actual distribution. Diagnoses are performed independently, and participants are discouraged from discussing their diagnoses or viewing each other's annotations in order to maintain objectivity.

In addition to diagnostic accuracy, the time required to complete the annotation task is also considered. Each medical intern is asked to keep track of the time spent during the evaluation process. The average duration of completion across the five interns is recorded as 27.5 minutes for the test set. For reference, the ground truth annotations are provided by an orthopedic surgeon, who completed the task in approximately 10 minutes. The EfficientNet model, on the other hand, completed the same task in less than 20 seconds, which is 0.33 minutes, including pre-processing and mask generation. These results highlight a significant difference in efficiency between human annotators and the automated model, particularly in scenarios where rapid diagnostic support is essential (see Figure 11.1).

To assess the accuracy of the interns' bounding box annotations in comparison to the segmentation masks produced by the model, IoU is utilized as the primary evaluation metric. This metric quantifies the spatial overlap between each intern's predictions and the ground truth segmentation mask provided by the orthopedic surgeon.

The following criteria are adopted for interpreting IoU scores in both clinical and segmentation contexts. When the IoU score exceeds 0.7, the intern annotation is considered

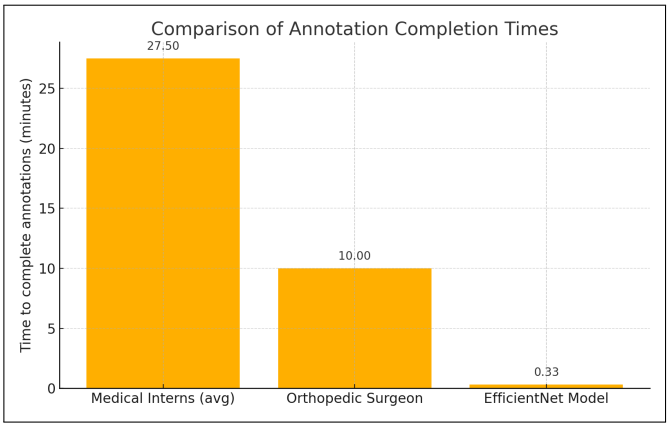


Figure 11.1. A bar chart comparing the completion times.

fully correct, indicating accurate localization of the fracture both from a medical and segmentation perspective. Scores between 0.3 and 0.7 are interpreted as partially correct in terms of segmentation, but still acceptable from a medical standpoint, as they reflect a correct diagnostic judgment despite incomplete localization. Figure 11.2 illustrates a score of 0.4, where the blue box shows the correct and complete detection of a fracture and the orange box only covers a *part* of the fracture. While the orange box indicates some overlap, it does not mark the complete area of fracture.

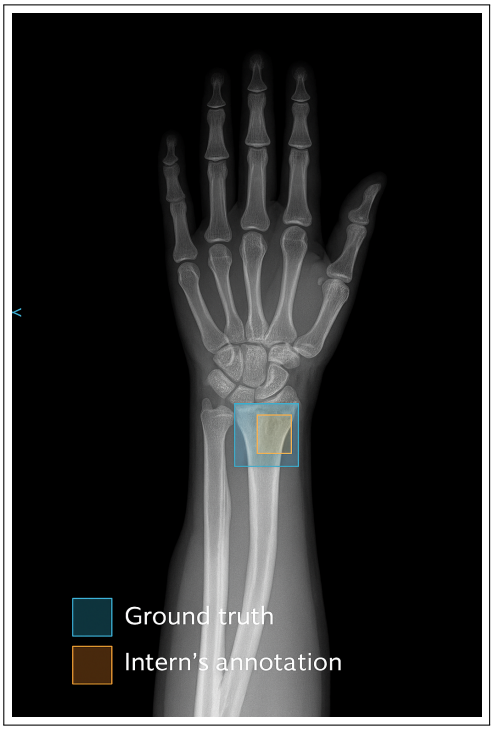


Figure 11.2. A overlap score of 0.4 on a sample radiograph.

Annotations with IoU values below 0.3 are regarded as incorrect from a segmentation perspective; however, a correct medical interpretation may still be acknowledged if the annotated region is near or adjacent to the actual fracture. Nevertheless, for the purposes of this study, the comparison prioritizes segmentation-based correctness, as the evaluation focuses on the model's ability to accurately localize fracture boundaries in relation to the provided ground truth.

To evaluate the overall diagnostic performance of the medical interns and enable a direct comparison with the segmentation model, the results of all 5 participants are aggregated and analyzed quantitatively, presented in Table 11.1. Each intern annotated the same 42 ground truth fractures, resulting in a total of 210 evaluated instances across the group. For each intern, the number of FPs, FNs, and partial detections are recorded. A partial detection is defined as a bounding box that correctly identifies the fracture but fails to encompass the entire region, based on the IoU-based thresholds described earlier. These partial annotations are conservatively counted as 0.5 TPs to reflect their partial segmentation accuracy while acknowledging their medical correctness.

Table 11.1. Intern detection summary

Intern	FP	FN	Partial Detections	Partial (counted as 0.5 TP)	TP
1	2	3	4	2.0	37.0
2	5	13	6	3.0	26.0
3	3	4	3	1.5	36.5
4	7	4	2	1.0	37.0
5	9	5	1	0.5	36.5

The first medical intern demonstrates generally consistent performance but exhibits notable issues in a subset of cases. A total of ten problematic instances are observed, comprising 6 AP views and four lateral views. Among the lateral images, 3 include annotations that cover only a portion of the fracture, resulting in segmentations that are considered partially correct; however, the fractures are medically identified. One additional lateral image is marked as an FN, with no annotation present despite the presence of a fracture (see Table 11.1).

The following describes the specific issues observed in the intern's AP view annotations. In one case, the radiograph contains two distinct fracture regions, but the intern correctly identifies only one, resulting in an incomplete annotation. Another case includes a false

positive, where the intern marks a region as fractured despite the absence of any fracture. A separate image contains both a false positive and a false negative: the intern incorrectly highlights the epiphyseal line, which is a normal anatomical feature in growing individuals that may appear radiolucent, as a fracture, while simultaneously missing the actual fracture elsewhere. One AP view results in a complete false negative, where the fracture is entirely missed. Finally, in one case, the annotation covers only part of the fracture region and is therefore considered partially correct from a segmentation standpoint.

The second intern encountered a higher number of problematic cases, with a total of twenty-two suboptimal annotations, particularly 11 in lateral views and 11 in AP views. Several of these cases involve both segmentation and diagnostic inaccuracies. In two lateral radiographs, the ground truths include both false positives and false negatives due to incorrect identification of anatomical structures that do not represent fractures. Multiple instances involve partially correct annotations where only a portion of the fracture is highlighted, indicating limited segmentation accuracy despite medically correct identification.

A total of 13 false negatives are observed across this intern's annotations, reflecting missed fractures in both views. Additionally, false positive annotations are recorded in 4 AP where regions without fractures are marked incorrectly. In another instance, both a false positive and a false negative are marked, suggesting diagnostic uncertainty. Overall, the second intern's annotations reflect a mix of partial segmentations and clinically incorrect predictions, resulting in reduced consistency in both localization and diagnosis.

The third intern presents a total of ten problematic cases, which include 3 FPs, 4 FNs, and 3 partial annotations. All 3 false positives occur in AP views, one of which involved the misinterpretation of the epiphyseal line as a fracture. In these instances, regions without actual fractures are incorrectly annotated, leading to medically incorrect diagnoses. Partial annotations are observed in one lateral and two AP views, where the fracture is identified but not fully enclosed, resulting in insufficient segmentation accuracy. Additionally, two AP views and two lateral views are marked as false negatives, with fractures entirely missed.

The fourth intern produces a total of 13 problematic annotations, consisting of 7 false positives, 4 false negatives, and 2 partial annotations. False positives dominate the error distribution, with 5 occurring in AP views and 2 in lateral views. In one AP image,

the epiphyseal line is annotated as a fracture, leading to both a false positive and a misinterpretation of normal anatomy. One AP image contains a false positive along with a partially correct annotation, while two additional AP views include false positives alone. False negatives occur in two AP views and two lateral views, where fractures are completely unmarked. Partial annotations appear once in each view type, reflecting situations where the intern identifies the correct area but fails to fully cover the fracture, resulting in partial segmentation accuracy. One lateral image includes both a false positive and a false negative, highlighting both over and under-detection within the same case.

The last intern has 15 erroneous cases, composed of 9 false positives, 5 false negatives, and one partial annotation. 6 of the false positives are observed in AP views, while 3 occur in lateral views. In one AP image, the epiphyseal line is once more mistaken for a fracture, contributing to diagnostic error. Two AP images are identified as false negatives, with actual fractures completely missed, while three additional false negatives are distributed across AP and lateral views. One AP image includes a partial annotation, where the fracture is recognized but not completely localized. The overall pattern in this intern's annotations suggests a tendency toward over-diagnosis, with a high false positive rate and several instances of incomplete or missed localization, particularly in AP views.

The final counts are summed across all interns to compute the total number of false positives, false negatives, partial detections, and true positives. These values are then converted into percentages to provide a normalized performance overview which is shown in Table 11.2. The resulting summary shows that, on average, the interns achieved a true positive rate of 82.3%, with 13.8% false negatives and 12.4% false positives. Partial detections constituted approximately 3.8% of all cases. This quantitative framework not only provides a structured basis for evaluating human diagnostic performance but also allows for a fair and consistent comparison against the segmentation model's outputs. The accompanying tables and charts derived from this analysis are used in the following sections to directly compare the interns' collective performance with that of the EfficientNet model.

In Figure 11.3, the bar chart and pie chart for the interns' collective performances are depicted. While the both the bar and pie charts show that TPs make up the majority of the interns' outcomes, the percentage of FNs is significant. The high percentage of FPs suggests that

Table 11.2. Cumulative intern detection performance

Metric	Percentage (%)
FP	12.4
FN	13.8
Partial	3.8
TP	82.3

sometimes interns might over-diagnose, possibly due to uncertainty or misinterpretation of normal anatomical features such as epiphyseal lines. Partial detections, while medically correct, reflect challenges in precisely localizing the full extent of the fracture which is an important metric for segmentation accuracy.

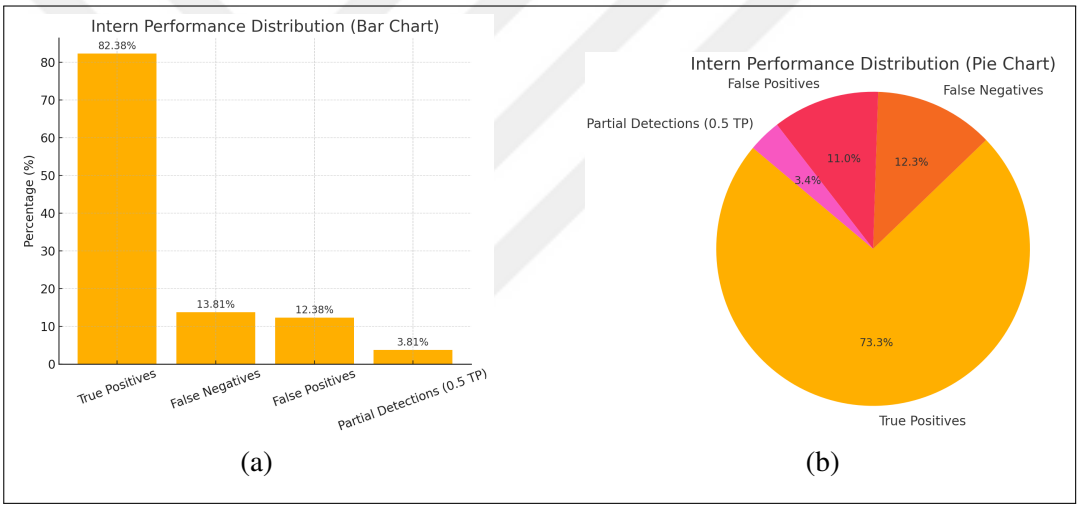


Figure 11.3. (a) Bar chart, (b) pie chart of collective intern performance.

Furthermore, while the pie chart in Figure 11.3b visually reinforces the dominance of true positives, it also shows that nearly one-fifth of all predictions are either missed or inaccurately identified. The clear segmentation between categories makes it possible to observe that over 17% of the responses (false positives + partials) indicate localization issues, even if the diagnostic decision was partially correct.

Lastly, an additional observation can be made based on the collective performance of the interns. It can be stated that there is a relationship between time spent on the task and overall diagnostic performance such that interns who documented longer completion times for the test set generally produce more accurate results, with fewer FNs and a higher number of correctly localized fractures. This trend suggests that deliberate and careful image review

contributes positively to diagnostic accuracy, particularly when precision in localization is required. While the annotation task was not time-restricted, the variation in completion times may reflect differences in approach, with more methodical evaluations resulting in improved performance.

In contrast, the EfficientNet segmentation model offers a consistent and rapid alternative, delivering predictions across all test images in under 20 seconds without variability in performance. All predictions are made at a confidence threshold of 0.9. On the hospital test set, the EfficientNet segmentation model demonstrates a high level of accuracy and consistency in localizing DRFs. Among the 40 test radiographs, the model produces only a single FN and 5 FPs in AP view. Of the 6 incorrect predictions in AP view, 2 false positives result from the model misinterpreting the epiphyseal line as a fracture, an error also observed among multiple interns. One false positive is caused by overlapping bone structures that create a visual pattern resembling a fracture. The remaining 2 occur in areas with ambiguous cortical irregularities, where no actual fractures are present. Aside from these cases, the model accurately identifies and segments all remaining fracture regions. Notably, it achieves perfect performance on lateral view radiographs, correctly segmenting every fracture without any false positives or false negatives.

A side-by-side analysis of the average detection performance of the 5 medical interns and the EfficientNet model is presented in Table 11.3. The EfficientNet demonstrates a clear advantage across multiple dimensions. From an accuracy perspective, the model achieves a true positive rate of 97.62%, significantly higher than the 82.38% observed among the interns. The false negative rate, which reflects missed fractures, is markedly lower for the model at 2.38%, compared to 13.81% for the interns. This indicates the model's strong ability to consistently detect fractures that are often overlooked by human observers. Additionally, while both the model and the interns show similar rates of false positives, the model makes these errors in more explainable contexts without over-marking unrelated regions.

Table 11.3. Interns versus model performance

	FPs (%)	**FNs (%)**	**Partial (%)**	**TPs (%)**	**Avg. Time per Img (s)**
Interns	12.4	13.8	3.8	82.3	42
EfficientNet	12	2.4	0.0	97.6	0.5

Partial detections, which occur when the annotated region only partially overlaps with the actual fracture, are absent in the model's predictions but account for 3.8% of the interns' outcomes. This reflects the model's strength in localizing fractures more completely and precisely, a result of its pixel-level segmentation capability.

Figure 11.4 provides the visual performance comparison of the interns and the EfficientNet model, clearly highlighting the model's superior accuracy, especially in reducing FNs and partial detections, as well as its significant advantage in speed.

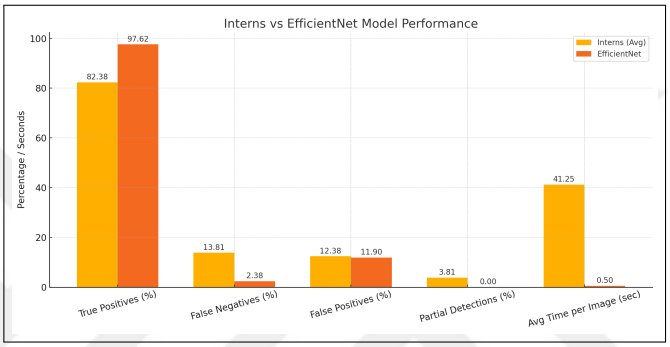


Figure 11.4. Bar chart comparing the performance of the model across interns.

While percentage-based comparisons offer valuable insight into the diagnostic tendencies of both the interns and the EfficientNet model, a more structured evaluation requires the use of standard performance metrics. To enable this, TPs, FPs, and FNs are used to calculate precision, recall, and F1-Score (see Table 11.4).

Table 11.4. Comparison metrics of interns versus EfficientNet model

	Precision	Recall	F1-Score
Interns	0.86	0.85	0.86
EfficientNet	0.9	0.98	0.93

The calculated evaluation metrics further emphasize the differences in diagnostic performance between the medical interns and the EfficientNet segmentation model. The average precision achieved by the interns is 0.86, which is lower than the model's precision of 0.9. This indicates that the model generates fewer FP predictions relative to its correct detections, reflecting more selective and consistent identification of actual fracture regions.

The most notable difference appears in the recall scores. The model achieves a recall of

0.98, whereas the interns reach 0.85. This gap reflects the model's superior ability to detect nearly all true fracture cases, minimizing missed detections. In clinical practice, this is particularly important, as undetected fractures can lead to delayed or inappropriate treatment. The F1-score, which provides a balanced measure of precision and recall, is also higher for the model compared to the interns. This reinforces the model's overall advantage in both correctly identifying and fully localizing fractures. Together, these metrics suggest that the EfficientNet model not only matches but exceeds the diagnostic performance of the interns, particularly in reducing overlooked fractures while maintaining high specificity.

The combined evaluation of diagnostic accuracy and efficiency reveals that the EfficientNet segmentation model consistently outperforms the group of medical interns across all key performance metrics. Furthermore, the model not only demonstrates a high degree of accuracy but also maintains a strong balance between sensitivity and specificity. In contrast, the interns collectively achieved lower scores across all metrics, which highlights a greater tendency to miss fractures. This discrepancy is clinically significant, as missed diagnoses can directly impact patient care.

Notably, the model did not miss a single fracture in any of the lateral view radiographs, whereas the interns frequently struggled with these images, contributing to a substantial portion of their false negatives. The model's ability to detect all lateral-view fractures suggests a strong capacity to generalize across different anatomical projections, which is particularly valuable in clinical practice where diagnostic accuracy must be maintained across variable imaging conditions.

These findings indicate that the EfficientNet model has reached a level of performance that is comparable to what would typically be expected from trained orthopedic specialists. While the interns' performance reflects the diagnostic capabilities of near-graduation medical students, the model offers greater consistency, precision, and speed, generating predictions in under one second per image. This combination of accuracy and efficiency positions the model as a promising tool for integration into clinical workflows, particularly as a decision support system in emergency or high-throughput radiology settings.

12. TECHNICAL DISCUSSION: INTERPRETABILITY VS EXPLAINABILITY

In AI, particularly within critical domains like medical diagnostics, the terms interpretability and explainability refer to distinct yet complementary attributes of model transparency. Interpretability concerns the inherent transparency of a model, allowing the user to comprehend the direct cause-and-effect relationships between inputs and outputs. The internal mechanisms of an interpretable model are discernible, enabling users to understand how a decision is made [99]. In contrast, explainability refers to techniques used to present the reasoning of a complex, often black-box, model in a human-understandable way. While explainability aims to answer why a model makes a specific prediction, interpretability focuses on understanding how the model works internally [99].

The prototype application enhances both interpretability and explainability through several design choices and output mechanisms. Firstly, radiographic images are displayed side by side after prediction, enabling clinicians to evaluate the model's decisions in real time and visually compare the original image with the predicted output. Secondly, the use of pixel-wise segmentation masks, rather than simple binary classifications, allows clinicians to assess the precise position, shape, and extent of a fracture. To further improve visual clarity, colormapped overlays are applied, helping the predicted fractures stand out against the background of the radiograph. In addition, a confidence score is presented alongside each prediction, informing the clinician of the model's certainty. A low confidence score may prompt further scrutiny or encourage a second opinion. Lastly, the CNN backbone of EfficientNet is designed to preserve detailed spatial features, resulting in precise masks that align closely with human perception of anatomical structures. Collectively, these measures improve the comprehensibility and clinical usability of the model's outputs.

To verify the interpretability and explainability of the model and to confirm the soundness of its decision-making process, Grad-CAM [100] is employed. Grad-CAM is a visualization technique that highlights the regions of an input image that are most influential in a CNN's prediction. It works by computing the gradients of the target class with respect to the final convolutional layer, producing a heatmap that reflects the areas the model focuses on when

making its decision.

Figure 12.1a illustrates the model's fracture prediction. In contrast, Figure 12.1b displays a Grad-CAM heatmap, revealing the regions that most strongly influenced the model's decision. The overlap between the heatmap and the predicted fracture location proves that the model's attention is focused on clinically meaningful areas, reinforcing the reliability of its decision-making process.

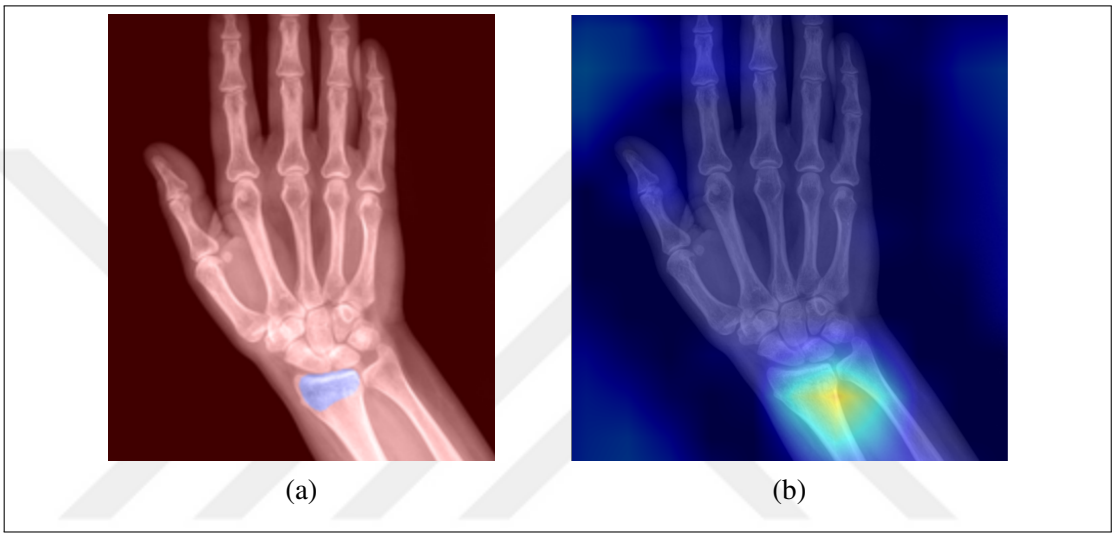


Figure 12.1. (a) Model's direct prediction, (b) Grad-CAM mapping of the prediction.

13. CONCLUSIONS

This study introduces a deep learning-based diagnostic system designed to detect and localize DRFs on wrist radiographs with high accuracy and efficiency. The primary aim was to develop a reliable and explainable system that could be feasibly integrated into clinical workflows and serve as a supportive tool for medical interns and students in radiology and emergency settings. To achieve this, a structured methodology was followed, beginning with the training and evaluation of several state-of-the-art computer vision models, including both detection and segmentation architectures. Models such as Faster R-CNN, YOLOv8, YOLOv11, Mask R-CNN, and EfficientNet were assessed thoroughly using standard metrics on a labeled dataset, with performance compared across multiple configurations and confidence thresholds.

Following the comparative evaluations and real-world testing, several critical findings were revealed. Among the object detection models, YOLOv11l emerged as the top performer, achieving the highest precision (96.5%), recall (95.7%), and F1-score (96.1%) on the real-world test set of 100 radiographs containing 116 ground truth fractures. This model not only minimized false positives and false negatives better than any other tested model but also demonstrated the highest accuracy (92.5%) while maintaining a reasonable inference time of 18.3 milliseconds per image. In contrast, the traditional region-based Faster R-CNN model achieved a recall of 87.3% and F1-score of 83.8%, but suffered from significant localization errors and high false positive counts, especially in radiographs with casts or surgical hardware.

The real-world testing phase underscored the resilience of YOLO-based detectors to complex cases, such as radiographs containing casts, surgical implants, or two-view wrist projections. For instance, YOLOv8m correctly identified 3 out of 5 casted fractures missed by Faster R-CNN and successfully detected a fracture in a low-contrast two-view radiograph—a case where Faster R-CNN failed. Likewise, YOLOv11m improved localization over earlier YOLO models, successfully detecting previously misclassified fractures, including those obscured by a cast or fragmented in nature.

The segmentation models were similarly evaluated, with EfficientNet demonstrating the most robust performance. It achieved a perfect recall of 100%, precision of 97%, and F1-score

of 98% on the real-world test set, successfully segmenting all fractures, including subtle and lateral-view cases, with only 3 false positives. This performance was superior to all YOLO-based segmentation models and Mask R-CNN, both of which exhibited higher false negative rates. For instance, YOLOv8l missed 25 out of 102 test fractures, and Mask R-CNN produced 10 false negatives and several false segmentations.

The system fully meets the functional requirements outlined in Chapter 4. It supported preprocessing, multi-model training, standardized evaluation using metrics like mAP, precision, recall, and F1-score, and storage of logs and checkpoints. All models were tested on a surgeon-annotated dataset, with detailed analysis of false positives and negatives. Predictions were visualized, and model performance was compared to medical interns based on accuracy, timing, and error types. Orthopedic surgeons also reviewed outputs and confirmed EfficientNet's clinical value.

For the Python-based application, all specified functionalities were implemented. Users can upload images, trigger inference with the best-performing model, view original and predicted outputs side-by-side, observe inference time and confidence score, save results, and submit feedback. Feedback can be submitted before or after prediction, and is stored as a log file along with metadata such as timestamp and image ID.

From a non-functional standpoint, the models exceeded the real-time processing requirement: all inferences were completed in under 1 second, well below the 5 second threshold. The feedback submission can be completed in 3 clicks, surpassing the original requirement of 5. Furthermore, the submission can also be completed under 15 seconds, depending on the feedback length. Fernet-based AES encryption was applied to all feedback logs, ensuring secure storage and restricted access. Last but not least, the application handled various image formats (JPEG, PNG, HEIC). However, formal usability validation of the application interface remains pending and is included as part of the future work.

To validate the clinical relevance of the system, an additional evaluation was conducted using a dataset of 40 wrist radiographs, independently collected from an orthopedic hospital and annotated by orthopedic surgeons. The EfficientNet model's performance was compared to that of 5 sixth-year medical interns with similar clinical backgrounds. Results show that the model consistently outperforms the interns across all key evaluation metrics. It achieved a

precision of 0.89, recall of 0.98, and F1-score of 0.93 while maintaining an average prediction time of only 0.5 seconds per image. The model successfully detected all fracture instances in lateral view radiographs, which was an area where the interns demonstrated noticeable difficulty. Additionally, the model showed greater consistency in fully localizing fractures, whereas interns frequently submitted incomplete or partial annotations.

Following the previous phases, the best performing model of all trained models, EfficientNet, was deployed within a custom-developed Python application. The application includes functionalities for image upload, automatic fracture segmentation, and optional user feedback collection, with predictions visualized as overlaid segmentation masks. This implementation is designed not only to support diagnostic decision-making but also to offer an interactive learning environment for trainees. The application can process radiographs quickly, producing high-resolution predictions while allowing users to compare model outputs with their own diagnostic interpretations.

In summary, the proposed system demonstrates strong potential for real-world integration as both a clinical support tool and an educational resource. The combination of high segmentation accuracy, minimal false negatives, rapid inference time, and an intuitive software interface positions the system as a valuable contribution to fracture detection workflows and medical training environments alike.

While the proposed system demonstrates promising results in both diagnostic accuracy and application readiness, several directions remain open for future development and clinical integration. As a next step, formal usability and interface evaluations will be conducted with practicing clinicians, including orthopedic surgeons, radiologists, and emergency medicine specialists. These studies will help assess the system's user interface (UI) and user experience (UX), particularly in terms of clarity, interpretability of results, and integration into routine workflows. Feedback from these evaluations will guide refinements to the interface and interaction design to ensure the application aligns with real-world diagnostic needs.

Another critical avenue of expansion involves the integration of the system with hospital infrastructures, such as Picture Archiving and Communication Systems (PACS) and Radiology Information Systems (RIS). Enabling seamless image retrieval and automated result storage would significantly enhance the system's usability in clinical environments and

reduce manual overhead. Additionally, the model's output behavior could be further adapted through an adjustable confidence threshold feature, allowing clinicians to tailor prediction sensitivity based on specific use cases—for instance, prioritizing sensitivity in screening settings and specificity in confirmatory stages.

From an educational perspective, the system could be extended to include a multi-user feedback tracking module and a dedicated training mode. The feedback mechanism would support timestamped diagnostic input from users and allow for post-evaluation review. A learning mode, where users can test their diagnostic skills against model predictions and later reveal ground truth annotations, would provide valuable educational support for medical students and residents.

To further validate the clinical utility of the system, comparative performance studies involving radiology residents and board-certified specialists can be conducted. These studies would offer a broader understanding of how the model performs relative to increasing levels of clinical expertise. Lastly, the core methodology may be adapted to detect other fracture types or anatomical regions. By retraining the segmentation model on appropriate datasets, the system could potentially support a wider range of musculoskeletal diagnostic tasks.

REFERENCES

1. University of West Florida. Body Terminology. Available from: https://pressbooks.uwf.edu/medicalterminology/chapter/body-terminology/. Accessed: 2025-04-09.

2. Mulders M, Rikli D, Goslings J, Schep N. Classification and treatment of distal radius fractures: a survey among orthopaedic trauma surgeons and residents. *European Journal of Trauma and Emergency Surgery*. 2017;43:239-48.

3. Mauck BM, Swigler CW. Evidence-based review of distal radius fractures. *Orthopedic Clinics*. 2018;49(2):211-22.

4. Azad A, Kang HP, Alluri RK, Vakhshori V, Kay HF, Ghiassi A. Epidemiological and treatment trends of distal radius fractures across multiple age groups. *Journal of wrist surgery*. 2019;8(04):305-11.

5. Colles A. On the fracture of the carpal extremity of the radius [1814]. *Injury*. 1970;2(1):48-50.

6. Tang A. Anatomy, shoulder and upper limb, hand carpal bones. Available from: https://www.ncbi.nlm.nih.gov/books/NBK535382/#:~:text=Introduction, Radiograph%20of%20the%20Wrist%20Joint).

7. Chung KC, Spilson SV. The frequency and epidemiology of hand and forearm fractures in the United States. *The Journal of hand surgery*. 2001;26(5):908-15.

8. Clement N, Duckworth A, Wickramasinghe N, Court-Brown C, McQueen M. Does socioeconomic status influence the epidemiology and outcome of distal radial fractures in adults? *European Journal of Orthopaedic Surgery & Traumatology*. 2017;27:1075-82.

9. Turner RG, Faber KJ, Athwal GS. Complications of distal radius fractures. *Orthopedic Clinics of North America*. 2007;38(2):217-28.

10. Flinkkilä T, Sirniö K, Hippi M, Hartonen S, Ruuhela R, Ohtonen P, et al. Epidemiology and seasonal variation of distal radius fractures in Oulu, Finland. *Osteoporosis*

international. 2011;22:2307-12.

11. Dicker RC, Coronado F, Koo D, Parrish RG. Principles of epidemiology in public health practice; an introduction to applied epidemiology and biostatistics. 2006.

12. Brogren E, Petranek M, Atroshi I. Incidence and characteristics of distal radius fractures in a southern Swedish region. *BMC musculoskeletal disorders*. 2007;8:1-8.

13. Plant C, Hickson C, Hedley H, Parsons N, Costa M. Is it time to revisit the AO classification of fractures of the distal radius?: Inter-and intra-observer reliability of the AO classification. *The bone & joint journal*. 2015;97(6):818-23.

14. Scarfe WC, Farman AG. What is cone-beam CT and how does it work? *Dental Clinics of North America*. 2008;52(4):707-30.

15. Kabaliuk N, Nejati A, Loch C, Schwass D, Cater J, Jermy M. Strategies for segmenting the upper airway in cone-beam computed tomography (CBCT) data. 2017.

16. Grunz JP, Jordan MC, Schmitt R, Luetkens KS, Huflage H, Meffert RH, et al. Gantry-free high-resolution cone-beam CT: efficacy for distal radius and scaphoid fracture detection and characterization. *Academic Radiology*. 2023;30(7):1358-66.

17. Moore CL, Copel JA. Point-of-care ultrasonography. *New England Journal of Medicine*. 2011;364(8):749-57.

18. Kozaci N, Ay MO, Akcimen M, Turhan G, Sasmaz I, Turhan S, et al. Evaluation of the effectiveness of bedside point-of-care ultrasound in the diagnosis and management of distal radius fractures. *The American journal of emergency medicine*. 2015;33(1):67-71.

19. Nissen-Lie H. Fracture radii “typica”. *Nord Med*. 1939;1:293-303.

20. Solgaard S. Classification of distal radius fractures. *Acta Orthopaedica Scandinavica*. 1985;56(3):249-52.

21. GARTLAND JR JJ, Werley CW. EVALUATION OF HEALED COLLES’FRACTURES. *JBJS*. 1951;33(4):895-907.

22. Lidström A. Fractures of the distal end of the radius: a clinical and statistical study of

end results. *Acta Orthopaedica Scandinavica*. 1959;30(sup41):1-118.

23. OLDER TM, Stabler EV, CASSEBAUM WH. Colles fracture: evaluation and selection of therapy. *Journal of Trauma and Acute Care Surgery*. 1965;5(4):469-76.

24. Frykman G. Fractures of the distal radius including sequelae, shoulder-hand-finger syndrome, disturbance in the distal radio-ulnar joint and impairment of nerve function; A clinical and experimental study. *Acta orthop Scand Suppl*. 1967;108:14-31.

25. Missakian ML, Cooney WP, Amadio PC, Glidewell HL. Open reduction and internal fixation for distal radius fractures. *The Journal of hand surgery*. 1992;17(4):745-55.

26. Green DP, et al. *Rockwood and Green's fractures in adults*. vol. 1. Lippincott Williams & Wilkins; 2010.

27. Abramo A, Kopylov P, Tägil M. Evaluation of a treatment protocol in distal radius fractures. *Acta orthopaedica*. 2008;79(3):376-85.

28. Gummesson C, Atroshi I, Ekdahl C. The disabilities of the arm, shoulder and hand (DASH) outcome questionnaire: longitudinal construct validity and measuring self-rated health change after surgery. *BMC musculoskeletal disorders*. 2003;4:1-6.

29. Trumble TE, Schmitt SR, Vedder NB. Factors affecting functional outcome of displaced intra-articular distal radius fractures. *The Journal of hand surgery*. 1994;19(2):325-40.

30. Ring D, Jupiter JB. Treatment of osteoporotic distal radius fractures. *Osteoporosis international*. 2005;16:S80-4.

31. Cherubino P, Bini A, Marcolli D. Management of distal radius fractures: treatment protocol and functional results. *Injury*. 2010;41(11):1120-6.

32. Rhee S, Kim J. Distal radius fracture metaphyseal comminution: A new radiographic parameter for quantifying, the metaphyseal collapse ratio (MCR). *Orthopaedics & Traumatology: Surgery & Research*. 2013;99(6):713-8.

33. Perugia D, Guzzini M, Civitenga C, Guidi M, Dominedò C, Fontana D, et al. Is it really necessary to restore radial anatomic parameters after distal radius fractures? *Injury*. 2014;45:S21-6.

34. Augat P, Iida H, Jiang Y, Diao E, Genant HK. Distal radius fractures: mechanisms of injury and strength prediction by bone mineral assessment. *Journal of Orthopaedic Research*. 1998;16(5):629-35.

35. Adams JE. Quantitative computed tomography. *European journal of radiology*. 2009;71(3):415-24.

36. Blake GM, Fogelman I. Technical principles of dual energy x-ray absorptiometry. *Seminars in nuclear medicine*. vol. 27. Elsevier. 1997:210-28.

37. Van Leerdam R, Wijffels M, Reijnierse M, Stomp W, Krijnen P, Schipper I. The value of computed tomography in detecting distal radioulnar joint instability after a distal radius fracture. *Journal of Hand Surgery (European Volume)*. 2017;42(5):501-6.

38. Ganesh D, Zirgibel B, Koval K, et al. The detection of prominent hardware in volar locked plating of distal radius fractures: intraoperative fluoroscopy versus computed tomography. *Journal of Orthopaedic Trauma*. 2016;30(11):618-21.

39. Ozer K, Toker S. Dorsal tangential view of the wrist to detect screw penetration to the dorsal cortex of the distal radius after volar fixed-angle plating. SAGE Publications Sage CA: Los Angeles, CA. 2011.

40. Ko C, Baird M, Close M, Cassas KJ. The diagnostic accuracy of ultrasound in detecting distal radius fractures in a pediatric population. *Clinical Journal of Sport Medicine*. 2019;29(5):426-9.

41. Hacihaliloglu I, Abugharbieh R, Hodgson A, Rohling R. Bone segmentation and fracture detection in ultrasound using 3D local phase features. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008: 11th International Conference, New York, NY, USA, September 6-10, 2008, Proceedings, Part I 11*. Springer. 2008:287-95.

42. Hacihaliloglu I, Abugharbieh R, Hodgson AJ, Rohling RN, Guy P. Automatic bone localization and fracture detection from volumetric ultrasound images using 3-D local phase features. *Ultrasound in medicine & biology*. 2012;38(1):128-44.

43. Shen D, Wu G, Suk HI. Deep learning in medical image analysis. *Annual review of biomedical engineering*. 2017;19(1):221-48.

44. Lindsey R, Daluiski A, Chopra S, Lachapelle A, Mozer M, Sicular S, et al. Deep neural network improves fracture detection by clinicians. *Proceedings of the National Academy of Sciences*. 2018;115(45):11591-6.

45. Blüthgen C, Becker AS, de Martini IV, Meier A, Martini K, Frauenfelder T. Detection and localization of distal radius fractures: Deep learning system versus radiologists. *European journal of radiology*. 2020;126:108925.

46. Zhang J, Li Z, Lin H, Xue M, Wang H, Fang Y, et al. Deep learning assisted diagnosis system: improving the diagnostic accuracy of distal radius fractures. *Frontiers in Medicine*. 2023;10:1224489.

47. Hardalaç F, Uysal F, Peker O, Çiçeklidağ M, Tolunay T, Tokgöz N, et al. Fracture detection in wrist X-ray images using deep learning-based object detection models. *Sensors*. 2022;22(3):1285.

48. Thian YL, Li Y, Jagmohan P, Sia D, Chan VEY, Tan RT. Convolutional neural networks for automated fracture detection and localization on wrist radiographs. *Radiology: Artificial Intelligence*. 2019;1(1):e180001.

49. Yahalomi E, Chernofsky M, Werman M. Detection of distal radius fractures trained by a small set of X-ray images and Faster R-CNN. *Intelligent Computing: Proceedings of the 2019 Computing Conference, Volume 1*. Springer. 2019:971-81.

50. Tobler P, Cyriac J, Kovacs BK, Hofmann V, Sexauer R, Paciolla F, et al. AI-based detection and classification of distal radius fractures using low-effort data labeling: evaluation of applicability and effect of training set size. *European Radiology*. 2021;31:6816-24.

51. Suzuki T, Maki S, Yamazaki T, Wakita H, Toguchi Y, Horii M, et al. Detecting distal radial fractures from wrist radiographs using a deep convolutional neural network with an accuracy comparable to hand orthopedic surgeons. *Journal of digital imaging*. 2022;35(1):39-46.

52. Oka K, Shiode R, Yoshii Y, Tanaka H, Iwahashi T, Murase T. Artificial intelligence to diagnosis distal radius fracture using biplane plain X-rays. *Journal of orthopaedic surgery and research*. 2021;16:1-7.

53. Kim MW, Jung J, Park SJ, Park YS, Yi JH, Yang WS, et al. Application of convolutional neural networks for distal radio-ulnar fracture detection on plain radiographs in the emergency room. *Clinical and Experimental Emergency Medicine*. 2021;8(2):120.

54. Anttila TT, Karjalainen TV, Mäkelä TO, Waris EM, Lindfors NC, Leminen MM, et al. Detecting distal radius fractures using a segmentation-based deep learning model. *Journal of digital imaging*. 2023;36(2):679-87.

55. Min H, Rabi Y, Wadhawan A, Bourgeat P, Dowling J, White J, et al. Automatic classification of distal radius fracture using a two-stage ensemble deep learning framework. *Physical and engineering sciences in medicine*. 2023;46(2):877-86.

56. Raisuddin AM, Vaattovaara E, Nevalainen M, Nikki M, Järvenpää E, Makkonen K, et al. Critical evaluation of deep neural networks for wrist fracture detection. *Scientific reports*. 2021;11(1):6006.

57. Kim D, MacKinnon T. Artificial intelligence in fracture detection: transfer learning from deep convolutional neural networks. *Clinical radiology*. 2018;73(5):439-45.

58. Daluiski A, Hotchkiss RN, Hanel D, Chopra S, Lindsey R. A Convolutional Neural Network Automatically Detects and Localizes Fractures of the Distal Radius with Greater Accuracy than Inexperienced Clinicians: Level 2 Evidence. *Journal of Hand Surgery*. 2017;42(9):S20-1.

59. Knight J, Zhou Y, Keen C, Hareendranathan AR, Alves-Pereira F, Ghasseminia S, et al. 2D/3D ultrasound diagnosis of pediatric distal radius fractures by human readers vs artificial intelligence. *Scientific Reports*. 2023;13(1):14535.

60. X-rays. Available from: https://www.nibib.nih.gov/science-education/science-topics/x-rays.

61. Bhat AK, Kumar B, Acharya A. Radiographic imaging of the wrist. *Indian journal*

of plastic surgery: official publication of the Association of Plastic Surgeons of India. 2011;44(2):186.

62. Manaster BJ, May DA, Disler DG. *Musculoskeletal Imaging: The Requisites E-Book: The Core Requisites*. Elsevier Health Sciences; 2013.

63. Long BW, Rollins JH, Smith BJ. *Merrill's Atlas of Radiographic Positioning and Procedures E-Book: Volume 3*. Elsevier Health Sciences; 2018.

64. Wells PN. Ultrasound imaging. *Physics in medicine & biology*. 2006;51(13):R83.

65. Posadzy M, Desimpel J, Vanhoenacker F. Cone beam CT of the musculoskeletal system: clinical applications. *Insights into imaging*. 2018;9:35-45.

66. Labelbox [cited 2024 28 January]. Available from: https://labelbox.com/services/labeling/.

67. Roboflow [cited 2024 28 January]. Available from: https://roboflow.com/.

68. CVAT [cited 2024 28 January]. Available from: https://www.cvat.ai/.

69. Rectlabel - Ryo Kawamura [cited 2024 28 January]. Available from: https://rectlabel.com/.

70. He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*. 2017:2961-9.

71. Parker JA, Kenyon RV, Troxel DE. Comparison of interpolating methods for image resampling. *IEEE Transactions on medical imaging*. 1983;2(1):31-9.

72. Han D. Comparison of commonly used image interpolation methods. *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*. Atlantis Press. 2013:1556-9.

73. Lehmann TM, Gonner C, Spitzer K. Survey: Interpolation methods in medical image processing. *IEEE transactions on medical imaging*. 1999;18(11):1049-75.

74. Davies ER. *Machine vision: theory, algorithms, practicalities*. Elsevier; 2004.

75. Nixon MS, Aguado AS. Basic image processing operations. *Feature extraction & image processing for computer vision*. 2012:83-136.

76. Kim J, Hwang JI, Lee J. VR Color Picker: Three-Dimensional Color Selection Interfaces. *IEEE Access*. 2022;10:65809-24.

77. Saha S. A comprehensive guide to convolutional neural networks—the ELI5 way. *Towards data science*. 2018;15:15.

78. Yamashita R, Nishio M, Do RKG, Togashi K. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*. 2018;9:611-29.

79. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016:770-8.

80. Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. *International conference on machine learning*. PMLR. 2019:6105-14.

81. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015:234-41.

82. Girshick R. Fast r-cnn. *arXiv preprint arXiv:150408083*. 2015.

83. Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*. 2016;39(6):1137-49.

84. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016:779-88.

85. Redmon J, Farhadi A. YOLO9000: better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017:7263-71.

86. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft

coco: Common objects in context. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014:740-55.

87. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009:248-55.

88. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A. The pascal visual object classes (voc) challenge. *International journal of computer vision*. 2010;88:303-38.

89. Redmon J. Yolov3: An incremental improvement. *arXiv preprint arXiv:180402767*. 2018.

90. Bochkovskiy A, Wang CY, Liao HYM. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:200410934*. 2020.

91. Jocher G, Chaurasia A, Stoken A, Borovec J, NanoCode012, Kwon Y, et al. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. Zenodo. 2022. Available from: https://doi.org/10.5281/zenodo.7347926.

92. Jocher G, Qiu J, Chaurasia A. Ultralytics YOLO. 2023. Available from: https://github.com/ultralytics/ultralytics.

93. Jegham N, Koh CY, Abdelatti M, Hendawi A. Evaluating the evolution of yolo (you only look once) models: A comprehensive benchmark study of yolo11 and its predecessors. *arXiv preprint arXiv:241100201*. 2024.

94. Jocher G, Qiu J. Ultralytics YOLO11. 2024. Available from: https://github.com/ultralytics/ultralytics.

95. Selcuk B, Serif T. A comparison of YOLOv5 and YOLOv8 in the context of mobile UI detection. *International Conference on Mobile Web and Intelligent Information Systems*. Springer. 2023:161-74.

96. Selcuk B, Serif T. Brain tumor detection and localization with YOLOv8. *2023*

8th International Conference on Computer Science and Engineering (UBMK). IEEE. 2023:477-81.

97. Abedeen I, Rahman MA, Prottyasha FZ, Ahmed T, Chowdhury TM, Shatabda S. Fracatlas: A dataset for fracture classification, localization and segmentation of musculoskeletal radiographs. *Scientific Data*. 2023;10(1):521.

98. Nagy E, Janisch M, Hržić F, Sorantin E, Tschauner S. A pediatric wrist trauma X-ray dataset (GRAZPEDWRI-DX) for machine learning. *Scientific data*. 2022;9(1):222.

99. Linardatos P, Papastefanopoulos V, Kotsiantis S. Explainable ai: A review of machine learning interpretability methods. *Entropy*. 2020;23(1):18.

100. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE international conference on computer vision*. 2017:618-26.