T.C.
EGE ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

LİSANS BİTİRME PROJESİ SONUÇ RAPORU

Derin Öğrenme Yöntemiyle X-Ray Kemik Görüntüleri Üzerinde Kırık Tespiti

Detection of Fractures on X-Ray Bone Images with Deep Learning Methods

Emir KIVRAK

05200000484, 05200000484@ogrenci.ege.edu.tr, 0534 825 27 08

İbrahim Halil ATAY

05200000481, 05200000481@ogrenci.ege.edu.tr, 0531 324 23 32

Proje Danışmanı: Doç. Dr. Erkan Zeki ENGİN

Bornova, İZMİR, Haziran 2025

E.Ü. ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
LİSANS BİTİRME PROJESİ BİLGİ FORMU

1- Projenin Yapıldığı Dönem: 2024-2025	2- Rapor Tarihi: 20.06.2025

3- Projenin Başlangıç ve Bitiş Tarihleri: Ekim 2024 – Haziran 2025

4- Projenin Adı: Derin Öğrenme Yöntemiyle X-Ray Kemik Görüntüleri Üzerinde Kırık Tespiti
Detection of Fractures on X-Ray Bone Images with Deep Learning Methods

Projenin Toplam Maliyeti: 2480,40 TL

5. Proje çalışma takımınızı belirtiniz. (Birini işaretleyiniz.)

☐ Disiplin içi takım çalışması, ✓ Çok disiplinli takım çalışması, ☐ Bireysel çalışma

6- Projeyi gerçekleştirecek öğrenciler ve iletişim bilgileri (adres, e-posta, tel.)

Emir Kıvrak
Bornova, İzmir
emirkooads@gmail.com
+905348252708

İbrahim Halil Atay
Bornova, İzmir
ibrahimhalilatay123@gmail.com
+905313242332

7- Projenin Yürütüldüğü Kuruluş: Ege Üniversitesi, Mühendislik Fakültesi, Elektrik Elektronik Mühendisliği Bölümü

8- Destekleyen Kuruluş(ların) Adı, Adresi ve Destek Miktarı: TUBITAK (2209A): 9000 TL

9- Öz: Bu çalışmada, X-Ray kemik görüntüleri üzerinden küçük kırıkların tespitini sağlayan derin öğrenme modeli geliştirilmiştir.

Anahtar Kelimeler: Derin öğrenme, X-Ray, kırık, etiketleme, nesne algılama, konumlandırma

12- Danışmanının Öğretim Üyesi Adı/Soyadı ve Görüşü: Doç. Dr. Erkan Zeki Engin

13- Bölüm Kurulu Görüşü:

14- Projenin Başarı Durumu: Projenin Aldığı Not:

Önsöz

Yazılım, yapay zeka ve derin öğrenme alanındaki hızlı gelişmeler, tıp ve sağlık hizmetleri sektöründe devrim niteliğinde yenilikçi çözümler sunma fırsatlarını beraberinde getirmiştir. Bu çerçevede, "Derin Öğrenme Yöntemiyle X-Ray Kemik Görüntüleri Üzerinde Kırık Tespiti" başlıklı tez çalışmamız, kemik kırıklarının doğru ve hızlı bir şekilde tespit edilmesini sağlayan bir derin öğrenme modeli geliştirmeye odaklanmaktadır.

Bu çalışmanın temel amacı, derin öğrenme algoritmalarını kullanarak X-Ray kemik görüntülerinde kırıkları tespit eden bir sistem tasarlamaktır. Böylece, sağlık hizmetlerine erişimin zor olduğu bölgelerde dahi, kırıkların erken tespiti ve tedavi süreçlerinin hızlandırılması ve hekimlerin daha hızlı bir şekilde kırık tespiti yapabilmelerini sağlayacak bir araç sunulması amaçlanmaktadır. Bu tür bir uygulama, özellikle acil müdahale gerektiren durumlarda büyük bir kolaylık sağlayacaktır.

Proje ekibi olarak, tez çalışmasını beraber yürüttüğümüz, süreç boyunca yol gösterimiz olan ve bizlere desteğini eksik etmeyen çok değerli danışman hocamız Doç. Dr. Erkan Zeki Engin'e ve 2209-A Üniversite Öğrencileri Araştırma Projeleri Destekleme Programı kapsamında projemizi destekleyen TÜBİTAK'a en içten teşekkürlerimizi sunarız.

Sonuç olarak, bu ve bu tür çalışmaların sağlık alanındaki dijital dönüşüme katkı sağlamasını ve hastaların daha hızlı, doğru teşhisler alarak tedavi süreçlerinin iyileşmesini sağlamasını umuyoruz.

Saygılarımızla,

Emir KIVRAK & İbrahim Halil ATAY

İZMİR - 2025

ÖZET MAKALE

Detection of Fractures on X-Ray Bone Images with Deep Learning Methods

Erkan Zeki ENGİN
Department of Electrical & Electronics Engineering Faculty of Engineering, Ege University Izmir, Turkey
erkan.zeki.engin@ege.edu.tr

Emir KIVRAK
Department of Electrical & Electronics Engineering Faculty of Engineering, Ege University Izmir, Turkey
emirkooads@gmail.com

İbrahim Halil ATAY
Department of Electrical & Electronics Engineering Faculty of Engineering, Ege University Izmir, Turkey
ibrahimhalilatay123@gmail.com

Abstract- This study focuses on the detection of fractures in X-Ray bone images using deep learning based object detection methods. A dataset was constructed by combining publicly available fractured and healthy bone images, which were then annotated using bounding boxes. Four models; YOLO, RT-DETR, Faster R-CNN, and RetinaNet, were trained and evaluated under various augmentation scenarios. RT-DETR-X achieved the highest performance with an mAP@50 score of 72.9%. To make the system user friendly and accessible, a desktop interface was developed using the Flask framework. The proposed tool aims to assist medical professionals in making quick and accurate diagnoses, particularly in emergency care or resource-limited clinical environments.

Keywords: Deep learning, X-Ray, fracture, labeling, object detection, positioning

1. Introduction

Stress fractures occur as a result of repetitive movements and mechanical overloading of the bones. There are two known subtypes: fatigue fractures and insufficiency fractures. Pain in the affected bone typically increases with activity and decreases with rest. They are commonly observed in the tibia, pelvis, and foot bones. In the early stages, direct radiographic findings are usually normal. Diagnosis is based on a detailed medical history, physical examination, and advanced imaging techniques. Stress fractures are often overlooked and difficult to diagnose; therefore, if there is clinical suspicion, they should be considered in the differential diagnosis [1].

In the literature, there are fracture detection studies based on artificial intelligence, developed using datasets composed of both open-source and clinical bone images collected from various medical devices.

Olczak et al. (2017) conducted a study where a deep neural network was trained with X-Ray images of fractured bones from specific body parts. The dataset included approximately 250,000 images, 56% of which were fractured bones and 36% were ankle images. The study achieved over 90% accuracy in body part recognition and 83% in fracture detection [2].

Raghavendra et al. (2018) used a thoracolumbar dataset consisting of 700 fractured and 420 non-fractured CT images, where their proposed CNN model achieved a classification accuracy of 99.1% [3]. Tobler et al. (2021) classified fractures using ResNet18 on a dataset of 15,775 frontal and lateral X-Ray images, achieving 94% accuracy [4]. Uysal et al. (2021) applied 26 different deep learning-based classification procedures to shoulder bone X-Rays from the MURA dataset and later developed two ensemble learning models to further improve the results [5]. Thian et al. (2019) designed a CNN-based deep network for fracture detection and localization on wrist radiographs. Their study included 7,356 wrist X-Ray images, annotated by radiologists. 90% of the dataset was used for training and 10% for

validation. The chosen architecture was Inception-ResNet with Faster R-CNN [6].

Jain et al. (2024) prepared a database of 4,000 images through data augmentation applied to 320 stress fracture X-Ray images. Using this dataset, they trained deep networks with ResNet-50 and AlexNet. While ResNet-50 achieved an accuracy of 68.17%, AlexNet reached 94.03%. Based on AlexNet's strong performance, they developed an improved algorithm named iA-HLD, which achieved 97.95% accuracy after 20 epochs [7].

Hardalaç et al. (2022) compared 20 open-source deep learning-based object detection models. Among these, Dynamic R-CNN achieved 77.7%, RetinaNet 66.8%, and PAA 62.9% in terms of mAP@50 [8].

Wang et al. (2022) worked on detecting stress fractures in hand bones using three different object detection models with varying backbones. Their custom model, WrisNet, achieved 56.6% AP in the final stage [9].

Wang et al. (2024) conducted a study using AdvYOLO, an improved version of the YOLOv8 model, for fracture detection in wrist X-Ray images. In this study, the mAP@50 score was improved from 63.8% in their previous work to 68.7% [10].

2. System Introduction and Operating Principles

The flowchart used in the study is shown in Figure 1.

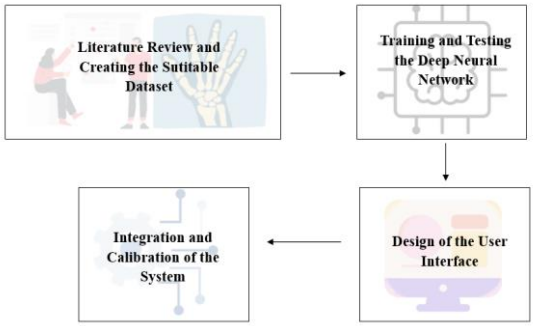


Figure 1 Flowchart of The System

2.1. Dataset Identification

A custom dataset was constructed by collecting approximately 250 X-Ray images from open-access sources such as FracAtlas, MURA, and Kaggle. These included both fractured and non-fractured bone images. Bounding box annotations were performed using the makesense.ai tool in YOLO format, later converted for compatibility with other object detection models. To improve model generalization and balance the dataset, data augmentation techniques were applied using the Albumentations library, increasing the dataset size to over 5,000 images. Additionally, healthy bone images with empty annotation files were included to prevent misclassification of anatomical structures such as joint gaps. The dataset was split into training (70%), validation (15%), and test (15%) sets.

2.2. Determination of the Environment for Model Development and Execution

Deep learning models typically operate on large datasets and therefore require significant computational power. As a result, advanced resources such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units) play a critical role in the training process. However, personal computers often lack the necessary processing capacity for such intensive tasks. At this point, Google Colab offers an accessible solution by providing powerful GPU resources through its free or Pro+ subscription tiers. Being a cloud based platform, Google Colab can be accessed from any device with an internet connection, making it highly convenient for team collaboration and cross-device development. Colab notebooks can easily be shared, which simplifies feedback collection and collaborative workflows. Moreover, Google Colab is fully compatible with popular deep learning libraries such as TensorFlow, PyTorch, Keras, Detectron2 and Ultralytics, allowing seamless integration with commonly used tools in model development. Its built-in integration with Google Drive further simplifies data storage and workflow management.

In conclusion, Google Colab was chosen in this project to provide the necessary computational capacity for model training and experimentation. The Cloud based nature of the platform contributed to a more efficient and flexible development process. To access more powerful hardware and ensure uninterrupted training sessions, a Google Colab Pro+ subscription was also purchased during later stages of the project.

2.3. Models

In this study, a range of state-of-the-art object detection models were explored to evaluate their effectiveness in identifying fractures in X-Ray bone images. The selected models; YOLO, RT-DETR, Faster R-CNN, and RetinaNet, were chosen based on their distinct architectural advantages, such as real time performance, transformer based design, and robustness in handling small object detection tasks. Each model was trained and assessed under various dataset configurations to determine the most suitable approach for the fracture detection task.

2.3.1 YOLO (You Only Look Once)

YOLO is a one-stage model family developed for real-time object detection. One of the most recent versions, YOLOv8, features a CSPDarknet-based backbone and compact C2f-structured blocks. The model operates using an anchor-free and decoupled head approach, separating classification and regression into distinct branches. With its low latency and high accuracy, the YOLO architecture offers a fast and effective solution, particularly in detecting small objects.

In the study, the object detection experiments initially began with the YOLO architecture, and YOLOv8 was selected as the preferred version. Training was conducted using various pretrained subversions of YOLOv8, including YOLOv8-S (Small), YOLOv8-M (Medium), YOLOv8-L (Large), and YOLOv8-X (Extra Large). The size of each model reflects its number of parameters, which, while potentially improving performance, also leads to slower training times. In the training and testing phases, an overall performance increase was observed with larger models, except for YOLOv8-L. Since YOLOv8-M, the predecessor of YOLOv8-L, achieved superior performance with a shorter training time, experiments using YOLOv8-L were discontinued.

2.3.2 RT-DETR (Real-Time Detection Transformer)

RT-DETR is an optimized version of the detection transformer architecture and follows a fully transformer based approach. The model typically employs ResNet or Swin Transformer as its backbone and uses a query-based decoder to directly generate bounding boxes without relying on anchor boxes or non-maximum suppression (NMS). Training is accelerated through techniques such as denoising training, and the model captures strong contextual relationships, offering deep spatial awareness, particularly in small object detection tasks.

For the training process, two pretrained versions of RT-DETR were utilized: RT-DETR-L (Large) and RT-DETR-X (Extra Large). The model size reflects the number of parameters, which can enhance accuracy but also result in longer training times.

Similar to YOLO, the RT-DETR deep network was implemented using the Ultralytics framework, and the training process employed identical hyperparameter configurations as those used in YOLO models.

2.3.3 Faster R-CNN

Faster R-CNN is a two-stage object detection model that first generates region proposals using a Region Proposal Network (RPN) and then classifies these regions to produce the final predictions. It typically employs a ResNet + Feature Pyramid Network (FPN) combination as its backbone. Although this modular architecture offers high accuracy, it results in slower inference times. The anchor

based structure and the need for fine-tuning of the RPN make it particularly sensitive and complex for small object detection tasks.

Unlike YOLO and RT-DETR, the training of the Faster R-CNN model in this project was carried out using the Detectron2 library, an advanced object detection framework developed by Facebook AI Research (FAIR) and built on PyTorch.

Initially, the YOLO format annotation files were converted to the COCO JSON format required by Detectron2. A custom Python script was written for this conversion process.

2.3.4 RetinaNet

RetinaNet is a one-stage object detection model that effectively addresses the class imbalance problem through the use of the Focal Loss function. It employs a ResNet + Feature Pyramid Network (FPN) combination as its backbone and utilizes multi-scale feature maps to detect objects at different. Thanks to its simple yet effective architecture, RetinaNet offers a balanced trade-off between accuracy and speed. However, when the dataset contains a high proportion of negative samples, the model may have a tendency to produce more false positives.

Unlike the previous three models, the Torchvision library was used for training the RetinaNet model. Torchvision is part of the PyTorch ecosystem and provides access to pretrained models, standard datasets, and data transformation tools.

Before training, the annotation files originally in YOLO format were converted to the CSV format required by Torchvision. A custom Python script was written to handle this conversion.

3. System Analysis Through Software-Based Simulations

In deep learning models, predictions are evaluated and classified using a Confusion Matrix, as shown in Table 1 below:

Table 1 Confusion Matrix

Predicted/Actual	Positive	Negative
Positive	TP	FP
Negative	FN	TN

The corresponding definitions in the context of this project are as follows:

- TP (True Positive): Correct detection of a fracture in a region that contains a fracture
- FP (False Positive): Incorrect detection of a fracture in a region that does not contain a fracture
- FN (False Negative): Failure to detect a fracture in a fractured region
- TN (True Negative): Correct classification of a healthy region with no fracture

Another important evaluation metric to mention is Intersection over Union (IoU); is a crucial metric used to evaluate the accuracy of object detection algorithms in deep learning. It is calculated by dividing the area of overlap between the predicted bounding box and the ground truth bounding box by the area of their union:

$$IoU = \frac{Intersection\ Area}{Union\ Area}$$

The general performance metrics used to evaluate the success of the deep learning models trained in this study are presented in Table 2.

Table 2 Performance Metrics Used

Metric	Definition	Formula	Interpretation
Precision	Indicates how many of the predicted positive results are actually correct. Focuses on reducing false positives.	$= \frac{TP}{TP + FP}$	A high value indicates that the model is good at avoiding false positives.
Recall	Indicates how well the model identifies all actual positive cases. Focuses on reducing false negatives.	$= \frac{TP}{TP + FN}$	A high value shows that the model is effective at detecting all target objects.
mAP@50	Mean Average Precision at an IoU threshold of 50%. The average precision across all classes at 50% overlap is used as the main accuracy indicator.	$= \frac{1}{N}\sum_{i=1}^{N} AP_i$	Used as the main indicator of overall performance. A higher value indicates both precision and recall.

3.1. Comparison of Performances

To assess the effectiveness of the developed system, four state-of-the-art object detection models; YOLOv8, RT-DETR, Faster R-CNN, and RetinaNet, were trained and tested on X-Ray bone images under four different dataset configurations. These dataset scenarios varied by the presence of data augmentation and the inclusion of both fractured and non-fractured images. Each model was evaluated using standard performance metrics including Precision, Recall, and mean Average Precision at IoU 50% (mAP@50).

YOLOv8, known for its real-time detection capabilities and anchor-free structure, performed well across all scenarios. Among its variants, YOLOv8-X showed the highest accuracy, particularly when trained on the augmented mixed dataset, achieving a mAP@50 of 71.5%. The model's balance between speed and accuracy, along with its decoupled head and CSPDarknet backbone, contributed to its strong performance, especially in detecting fractures located near bone edges.

Faster R-CNN, a two-stage detection framework, demonstrated stable results with relatively high precision. Its region proposal network (RPN) helped in accurately identifying candidate regions, but its longer inference time and sensitivity to anchor configuration reduced its effectiveness in real-time scenarios. The model performed best on datasets without augmentation, suggesting it benefited more from raw, unaltered image distributions.

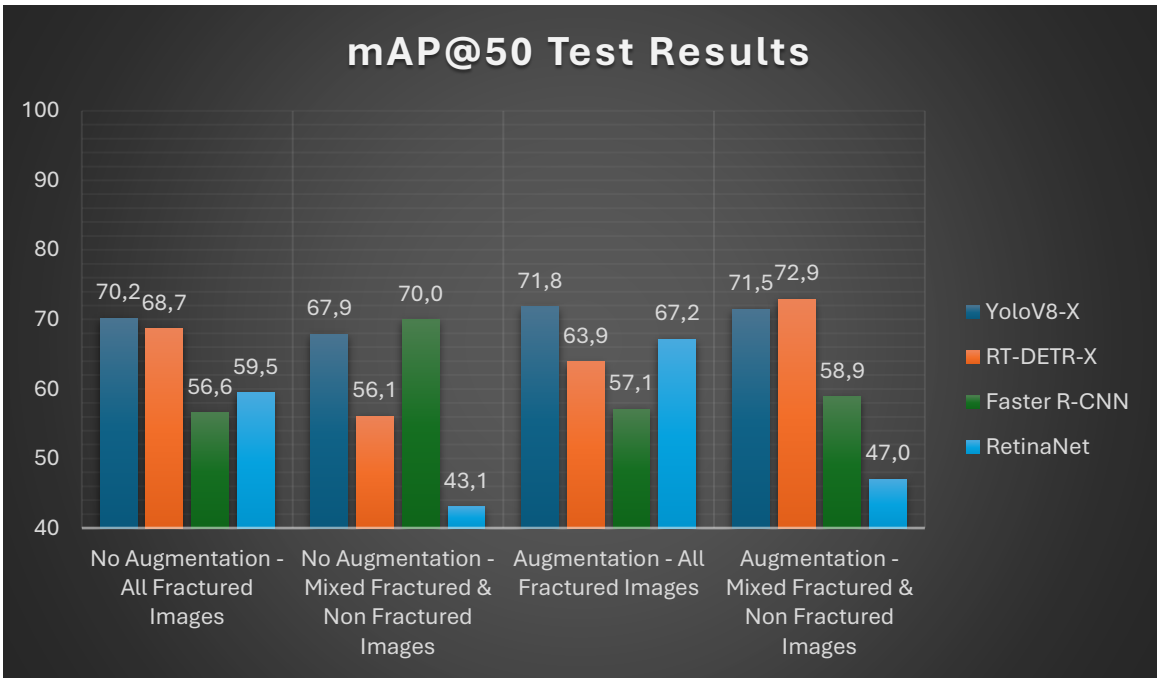
RetinaNet, which integrates focal loss to handle class imbalance, achieved moderate performance in datasets where negative samples (i.e., non-fractured images) were prevalent. However, it showed a tendency toward increased false positives under these conditions. Its single-stage nature and dense prediction heads made it faster than Faster R-CNN but less robust in distinguishing subtle fracture lines.

RT-DETR-X, a transformer-based model utilizing a query-based decoder, outperformed all other models in the study. It achieved a

mAP@50 of 72.9% when trained on the augmented mixed dataset. This superior performance can be attributed to its end-to-end transformer architecture, denoising training technique, and ability to model long-range spatial dependencies. Unlike traditional models, RT-DETR does not rely on anchor boxes or post-processing techniques like non-max suppression, which makes it especially effective in detecting small and ambiguous fracture patterns. The model was able to detect fractures near complex anatomical structures such as joints and overlapping tissues with higher accuracy than others, likely due to its contextual reasoning capabilities.

Overall performances of the models can be seen in Table 3.

Table 3 Overall Performances of the Models



3.2. Design of the User Interface & Integration and Calibration of the System

To ensure practical applicability and ease of use, the developed fracture detection model was integrated into a lightweight, desktop-based interface using the Flask micro web framework. Flask offered a flexible and efficient environment for handling key backend functionalities such as image uploading, model inference, and communication between components. Its simplicity and compatibility with Python made it an ideal choice for deploying the trained deep learning model in a user-accessible format.

The system enables users to upload X-Ray bone images, which are automatically processed by the underlying object detection model. The results are rendered visually on the interface with bounding boxes highlighting the detected fracture regions. This direct visual feedback helps bridge the gap between complex AI infrastructure and end-user interpretation.

The interface was designed with a focus on minimalism, clarity, and accessibility. Even users with limited technical background can operate the system seamlessly, as no manual preprocessing or configuration is required. Interactions are handled via a clean HTML layout that displays the uploaded image

alongside the detection results, allowing for an intuitive workflow.

By combining deep learning capabilities with an interactive, real-time interface, the project demonstrates how AI-based diagnostic tools can be adapted for clinical and research use. This integration not only enhances usability but also highlights the potential for deploying such models in real-world healthcare settings where speed, accuracy, and simplicity are essential.

Screenshots from the User Interface main page and model detection output can be seen in Figure 2 & 3.

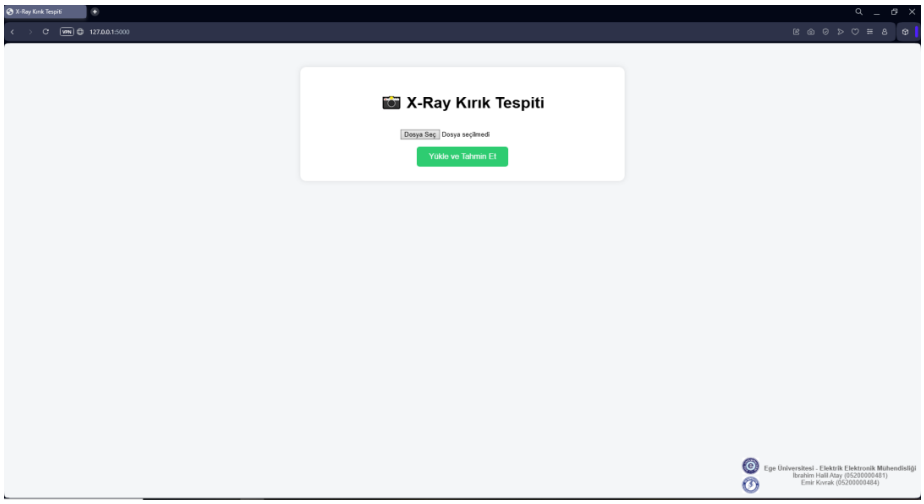


Figure 2 Main Page of the Designed User Interface

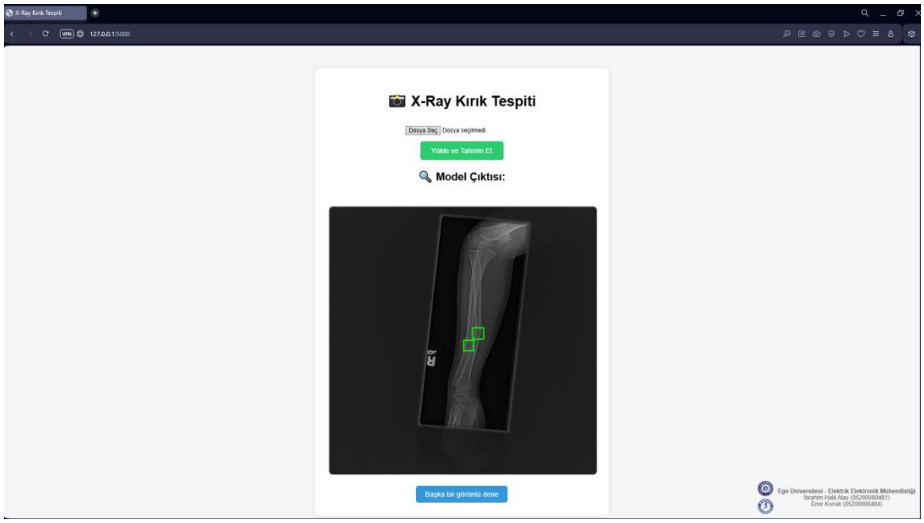


Figure 3 Model Detection Output on the Designed User Interface

4. Conclusion

In this study, a deep learning based object detection system was developed to identify and localize fractures in X-Ray bone images. Multiple state-of-the-art models, including; YOLOv8, RT-DETR, Faster R-CNN, and RetinaNet, were trained and evaluated across different dataset configurations. Among these, RT-DETR-X demonstrated the highest performance with a mAP@50 score of 72.9%

under an augmented and mixed dataset scenario.

To enhance usability, the best-performing model was integrated into a Flask-based desktop application, enabling end-users to upload X-Ray images and receive real-time visual outputs. This end-to-end system aims to support healthcare professionals by offering a fast, accessible, and accurate fracture detection tool.

Although the results are promising, certain challenges—such as dataset diversity, small object detection limitations, and class imbalance—remain to be addressed. Future work will focus on expanding the dataset, refining the models for different bone regions and fracture types, and validating the system in real clinical settings. Overall, this project highlights the practical potential of deep learning in orthopedic diagnostics and contributes to the growing field of AI-assisted medical imaging.

5. References

[1] Tuba Bahtiyarca, Z., Okan, S., Şahin, S. B., & Ocak, F. (2021). Stres Kırıklarına Yaklaşım: Üç Olgu Sunumu ve Literatürün Gözden Geçirilmesi. *Turkish Journal of Osteoporosis/Turk Osteoporoz Dergisi*, *27*(2).DOI: 10.4274/tod.galenos.2020.86648

[2] Olczak, J., Fahlberg, N., Maki, A., Razavian, A. S., Jilert, A., Stark, A., … Gordon, M. (2017). Artificial intelligence for analyzing orthopedic trauma radiographs. *Acta Orthopaedica*, *88*(6), 581–586. https://doi.org/10.1080/17453674.2017.1344459

[3] Raghavendra, U.; Bhat, N.S.; Gudigar, A.; Acharya, U.R. Automated system for the detection of thoracolumbar fractures using a CNN architecture. Future Gener. Comput. Syst. 2018, 85, 184–189. http://doi.org/10.1016/j.future.2018.03.023

[4] Tobler, P.; Cyriac, J.; Kovacs, B.K.; Hofmann, V.; Sexauer, R.; Paciolla, F.; Stieltjes, B.; Amsler, F.; Hirschmann, A. AI-based detection and classification of distal radius fractures using low-effort data labeling: Evaluation of applicability and effect of training set size. Eur. Radiol. 2021, 31, 6816–6824. http://doi.org/10.1007/s00330-021-07811-2

[5] Uysal, Fatih, Fırat Hardalaç, Ozan Peker, Tolga Tolunay, and Nil Tokgöz. 2021. "Classification of Shoulder X-ray Images with Deep Learning Ensemble Models" *Applied Sciences* 11, no. 6: 2723. https://doi.org/10.3390/app11062723

[6] Thian, Y.L.; Li, Y.; Jagmohan, P.; Sia, D.; Chan, V.E.Y.; Tan, R.T. Convolutional Neural Networks for Automated Fracture Detection and Localization on Wrist Radiographs. Radiol. Artif. Intell. 2019, 1, e180001. http://doi.org/10.1148/ryai.2019180001

[7] Jain, B., Malik, D., Jagota, G. *et al.* iA-HLD: an improved AlexNet for hairline fracture detection in orthopedic images. *Neural Comput & Applic* (2024). https://doi.org/10.1007/s00521-024-10348-2

[8] Hardalaç, F.; Uysal, F.; Peker, O.; Çiçeklidağ, M.; Tolunay, T.; Tokgöz, N.; Kutbay, U.; Demirciler, B.; Mert, F. Fracture Detection in Wrist X-ray Images Using Deep Learning-Based Object Detection Models. *Sensors* 2022, *22*, 1285. https://doi.org/10.3390/s22031285

[9] Wang, W., Huang, W., Lu, Q. *et al.* Attention mechanism-based deep learning method for hairline fracture detection in hand X-rays. *Neural Comput & Applic* 34, 18773–18785 (2022). https://doi.org/10.1007/s00521-022-07412-0

[10] Wang, R., Zhao, J., Liu, X. *et al.* AdvYOLO: Advanced YOLOv8 Application for Bone Pathology Localization and Classification in Wrist X-ray Images, 29 March 2024, PREPRINT (Version 1) available at Research Square https://doi.org/10.21203/rs.3.rs-4051336/v1

İçindekiler

Şekiller Tablosu

Tablolar Tablosu

1.ÖZ

Bu proje, derin öğrenme yöntemiyle X-Ray kemik görüntüleri üzerinden küçük kırıkların tespitini amaçlamaktadır. Daha önce yapılan benzer çalışmaların, derin öğrenme ile görüntü işleme tekniklerinden biri olan “Nesne Algılama" algoritmaları ile gerçekleştirilmesi yönündeki girişimler kısıtlıdır. Projede belirlenen veri setiyle eğitilen derin ağ aracılığıyla kırık tespit edilmesi istenen X-Ray kemik görüntüsü karşılaştırarak ilgili görüntüdeki kemikte kırık olup olmadığı ve nerede olduğu tespit edilmektedir. Projede öncelikle literatür taraması yapıldı ve uygun veri seti oluşturuldu. Oluşturulan bu veri setindeki görüntüler üzerinde kırıkların konumları etiketlendi. Sonrasında farklı derin ağların eğitilmesi ve performans testleri ile projeye en uygun ağ yapısı tespit edilmiştir. Derin ağın düzgün çalıştığı kesinleştikten sonra yerel bilgisayarlar üzerinde kullanılabilen bir arayüz aracılığı ile model tahminleri yaptırılıp çıktılar gösterilmiştir. Geliştirilen projenin ön sonuçlarından bir bildiri yapılması, bir ulusal makale yayınlanması, X-Ray kemik görüntülerinde kırık tespitinin maliyetinin azaltılması, insanların muayeneye teşvik edilmesi, hekimlere yardımcı bir uygulama geliştirilmesi, araştırmacı yetiştirilmesi, yeni projeler geliştirilmesi ve yüksek lisansa teşvik gibi yaygın etkileri bulunmaktadır. Proje, derin öğrenme ve ortopedi – travmotoloji sağlık hizmetleri alanlarının birlikte çalışabileceği potansiyelini göstermiş, ancak etkinliğin arttırılması ve yaygınlaştırılması için sürekli geliştirme gerekliliğini vurgulamıştır.

Anahtar Kelimeler: Derin öğrenme, X-Ray, kırık, etiketleme, nesne algılama, konumlandırma

Abstract: This project aims to detect small fractures in X-Ray bone images using deep learning methods. Previous similar studies have been limited in attempts to apply «Object Detection» algorithms, which are one of the image processing techniques in deep learning. The deep neural network trained with the selected dataset is used to compare X-Ray bone images and detect whether there is a fracture in the bone and, if so, where it is located. The project begins with a literature review and the creation of an appropriate dataset. The fracture locations in the images of this dataset were labeled. Subsequently, different deep networks were trained, and through performance testing, the most suitable network architecture for the project was determined. Once the deep network’s functionality is confirmed, predictions will be made through an interface that can be used on local computers, and the outputs will be displayed. The preliminary results of the developed project will have widespread effects, including publishing a report, a national article, reducing the cost of fracture detection in X-Ray bone images, encouraging people to seek medical examination, developing an auxiliary application for doctors, training researchers, creating new projects, and encouraging postgraduate studies. The project has demonstrated the potential for deep learning and orthopedics-traumatology healthcare fields to collaborate, but it has also highlighted the need for continuous development to increase effectiveness and broaden its implementation.

Keywords: Deep learning, X-Ray, fracture, labeling, object detection, positioning

2.GİRİŞ

Projenin Amacı

Bu projenin amacı, derin öğrenme temelli bir ağ yapısı tasarlayarak, tespiti insan gözü ile zor olan küçük kırıkların varlığını ve konumunu belirlemeye yardımcı olacak bir yazılım geliştirmektir.

Konu/Problem Tanımı

Kemik kırıkları, insan sağlığını doğrudan etkileyen, tedavi edilmediğinde ciddi komplikasyonlara yol açabilen yaygın bir sağlık problemidir. Özellikle trafik kazaları, spor yaralanmaları ve yaşlılık gibi faktörler nedeniyle, kemik kırıkları sıklıkla karşılaşılan bir durumdur. Geleneksel tıbbi yöntemlerle kırıkların tespiti, genellikle radyolojik görüntülerin uzmanlar tarafından manuel olarak incelenmesiyle yapılmaktadır. Ancak bu yöntem, zaman alıcıdır ve yanlış değerlendirmelerle birlikte tanı hatalarına yol açabilmektedir. Ayrıca, bu yöntemlerin uygulandığı sağlık kurumlarında sıklıkla yoğunluk yaşanmakta ve bu durum, hastaların tedavi süreçlerini olumsuz yönde etkileyebilmektedir.

Gelişen teknolojiyle birlikte, tıbbi görüntülerin analizi için kullanılan derin öğrenme yöntemleri, insan hata payını minimize ederek, daha hızlı ve doğru sonuçlar elde edilmesine olanak sağlamaktadır. Bu bağlamda, derin öğrenme algoritmalarının, kemik kırığı tespitinde kullanılması büyük bir potansiyele sahiptir. Derin öğrenme yöntemlerinin en büyük avantajı, büyük veri setlerinden otomatik olarak özellik çıkarımı yapabilmesidir. Bu sayede, uzmanlık gerektiren kararlar daha hızlı bir şekilde alınabilir ve tıbbi hatalar azaltılabilir.

Tez çalışmasında, derin öğrenme tabanlı nesne algılama (object detection) modelleri kullanarak, X-Ray kemik görüntülerinde kırıkların doğru bir şekilde tespit edilmesini amaçlamaktadır. Bu çalışmada, derin ağlar, büyük X-Ray veri setlerinden öğrenerek, kemiklerdeki kırıkları analiz etmekte ve konumlarını tespit etmektedir. Bu modelin temel avantajı, görüntülerdeki kırıkları hızlı bir şekilde tespit etmesi ve insan müdahalesine gerek kalmadan doğru sonuçlar üretmesidir.

Mevcut literatürde, derin öğrenme tabanlı nesne algılama modellerinin X-Ray kemik görüntülerinde uygulama alanı sınırlıdır. Bu alandaki çalışmalar genellikle temel kırık tespitiyle sınırlı kalmakta, tıbbi görüntüler üzerinden kırığın türü, konumu gibi daha detaylı verilerin analizi ihmal edilmektedir. Ayrıca, bu tür sistemlerin genellikle bilgisayar ortamında çalışabilir olması ve kullanıcılara pratikte kolaylık sağlayacak bir arayüzle entegrasyonunun eksikliği büyük bir sorun teşkil etmektedir.

Bu tezde, derin öğrenme yöntemiyle X-Ray kemik görüntülerindeki kırıkları tespit eden bir sistem geliştirilmiş ve bu sistem, bilgisayar ortamında açılabilir bir arayüz ile entegre edilmiştir. Bu arayüz, sağlık profesyonellerine X-Ray görüntülerini kolayca yükleyip analiz etmelerini sağlayacak basit ve kullanıcı dostu bir çözüm sunmaktadır. Böylece, sağlık hizmetleri sağlayıcılarının, özellikle acil servislerde ve yoğun kliniklerde, kırık tespiti yapabilmesi hızlanacak ve tedavi sürecindeki yanlışlıklar en aza indirgenecektir.

Sonuç olarak, bu çalışma yalnızca kemik kırıklarının doğru tespitini sağlamakla kalmayacak, aynı zamanda sağlık personellerine daha hızlı ve güvenilir sonuçlar sunarak, sağlık hizmetlerine erişim süreçlerini iyileştirmeyi amaçlamaktadır. Bilgisayar ortamında çalışan bir arayüz ile kullanıcı dostu bir çözüm sunulması, bu teknolojinin sağlık alanında yaygınlaşması için önemli bir adım olacaktır. Ayrıca, erken teşhis ve doğru tedavi süreci, kemik kırıklarıyla ilgili komplikasyonların önüne geçerek, daha sağlıklı bir toplum oluşturulmasına katkı sağlayacaktır.

Projenin Hedefleri

- **Yüksek Hassasiyet ve Keskinlik:** Küçük kırıkların var olup olmadığının %70 üzeri bir mAP50 değeri ile tespit edilmesi ve kırık olduğu tespit edilen görüntülerde kırık konumunun yüksek doğruluk ile belirlenmesi projenin ana hedefidir.
- **Veri Toplama ve Etik İlkeler:** Projede kullanılacak olan görüntülerin toplanması, saklanması ve işlenmesi sırasında gizlilik ve etik kurallarına uyulmalıdır. Olabildiğince yüksek kalitede ve sayıda X-Ray görüntüsü bulunmalıdır (en az 150 görüntü).
- **Asistan:** Proje, hekimlere kırığın var olup olmadığı ve yeri hakkında yardımcı olmayı amaçlamaktadır. Ayrıca yeni hekim adaylarına karşı da eğitici görev üstlenebilir.

Literatür Özeti

Stres kırıkları, kemiklerin tekrarlayan hareketleri ve mekanik olarak aşırı yüklenmesi sonucu ortaya çıkar. Bilinen iki alt tipi; yorgunluk kırıkları ve yetersizlik kırıklarıdır. Genellikle etkilenen kemikteki ağrı aktiviteyle artar, dinlenmeyle azalır. Sıklıkla tibia, pelvis ve ayak kemiklerinde görülürler. Erken dönemde direkt radyografi sonuçları genellikle normaldir. Tanı, ayrıntılı geçmiş anlatımı, fiziksel inceleme ve ileri görüntüleme yöntemlerine dayanır. Stres kırıkları sıklıkla gözden kaçırılır ve teşhis edilmesi zordur; bu nedenle klinik şüphe varsa ayırıcı tanıda göz önünde bulundurulmalıdır [1].

Literatürde, çeşitli tıbbi cihazlardan toplanan, hem açık kaynak hem de klinik kemik görüntülerinden oluşan veri setleriyle oluşturulan, yapay zekaya dayalı kırık kemik tespit çalışmaları yer almaktadır:

Olczak vd. (2017) yaptıkları çalışmada, vücudun belirli bölgelerinden çeşitli kırık kemiklerin X-Ray görüntüleri ile eğitilen bir derin ağ sayesinde kırık ve vücut bölgesi tespiti yapabilen bir çalışma gerçekleştirilmiştir. Yaklaşık olarak 250.000 görüntü içeren ve %56'sının kırık kemik görüntüsü, %36'sının da ayak bileği görüntüsü olduğu bir veri seti üzerinde çalışılmıştır. Vücut bölgesinin tanınmasında %90'ın üzerinde, kırığın tespitinde ise %83 oranında başarı elde edilmiştir [2].

Raghavendra vd. (2018) yaptıkları çalışmada 700 kırık ve 420 kırık olmayan tomografi görüntüsü içeren thoracolumber veri setinde önerilen CNN modeliyle %99,1'lik bir sınıflandırma doğruluğu elde edilmiştir [3]. Tobler vd. (2021) 15.775 frontal ve lateral röntgen film görüntüsünden oluşan veri setinde ResNet18 modeliyle kırık sınıflandırması gerçekleştirmiş ve %94'lük doğruluk oranı sağlanmıştır [4]. Uysal vd. (2021) kas-iskelet sistemi radyografisi (MURA) veri setindeki omuz kemiği X-Ray görüntülerinde kırık sınıfını belirlemek için 26 farklı derin öğrenmeye dayalı sınıflandırma prosedürü uyguladılar ve ardından sınıflandırmanın sonuçlarını daha da iyileştirmek için iki farklı topluluk öğrenme modeli geliştirmişlerdir [5]. Thian vd. (2019) bilek radyografilerinde kırık tespiti ve bölgelendirmesi için CNN tabanlı bir derin ağ tasarlamışlardır. Bu çalışma için 7356 adet bilek radyografi görüntüsü incelenmiştir. Kırıklar, radyolojistler tarafından işaretlenmiş ve veri setinin %90'ı yapay zekanın eğitilmesi ve %10'u ise validasyonu için kullanılmıştır. Inception-ResNet Faster R-CNN mimarisi derin öğrenme modeli olarak tercih edilmiştir [6].

Jain vd. (2024) yaptıkları çalışmada, 320 adet stres kırığı X-Ray görüntüsü üzerinden veri artırımı (data augmentation) yöntemlerini kullanarak 4000 görüntülük bir veri tabanı hazırlamışlardır. Bu veri tabanı üzerinden ResNet-50 ve AlexNet algoritmaları kullanılarak derin ağ eğitilmiştir. ResNet-50 kullanılarak yapılan çalışma ile %68,17'lik bir başarı oranı yakalanırken, AlexNet algoritması ile %94,03'lik bir başarı oranı yakalanmıştır. AlexNet'in gösterdiği bu başarılı performansa dayanarak

araştırmacılar bu algoritmayı geliştirerek iA-HLD adını koydukları algoritmayı oluşturmuşlardır. Geliştirilen bu iA-HLD algoritmasıyla derin ağ, 20 adım (epoch) eğitildiğinde %97,95'lik bir başarı oranı elde edilmiştir [7].

Hardalaç vd. (2022) yaptıkları çalışmada, 20 farklı açık kaynaklı derin öğrenme tabanlı nesne algılama modeli kullanarak modeller arasında kıyaslama gerçekleştirmişlerdir. Bu kıyaslamaya göre kullanılan modeller arasında Dynamic R-CNN modeli ile %77,7, RetinaNet modeli ile %66,8, PAA modeli ile %62,9 mAP50 değerlerine ulaşılmıştır [8].

Wang vd. (2022) yaptıkları çalışmada, el kemiklerine ait X-Ray görüntüleri üzerinden stres kırıklarının tespiti üzerine, çeşitli farklı backbone'lar ile 3 farklı nesne algılama modeli üzerinde çalışmıştır. Bu modellerden sonuncusu olan ve kendi geliştirdikleri WrisNet ile son aşamada %56.6 AP değerine ulaşılmıştır [9].

Wang vd. (2024) yaptıkları çalışmada, YOLOv8 modelinin geliştirilmiş bir versiyonu olan AdvYOLO algoritmasını kullanarak bilek kemiklerine ait X-Ray görüntüleri üzerinde kırık tespiti üzerine çalışmıştır. AdvYOLO algoritması ile yaptıkları çalışmada, mAP50 değerini önceki çalışmalarında elde ettikleri %63.8'den %68.7'ye çıkarmışlardır. [10]

3.GELİŞME

Teori

Derin öğrenme, yapay zeka alanında özellikle büyük veri setlerini analiz etme ve yüksek doğrulukla tahminlerde bulunma yeteneğiyle dikkat çeken bir yöntemdir. Derin öğrenme sistemleri, yapay sinir ağları (artificial neural networks) kullanarak veriyi öğrenir. Bu sistemler, insan beynine benzer şekilde çalışarak verilerdeki kalıpları öğrenir ve karmaşık görevleri yerine getirebilir. Derin öğrenmenin temeli, çok katmanlı ağlar kullanarak veri üzerinde soyutlama yapma yeteneğidir.

Bu çalışmada, Convolutional Neural Networks (CNN'ler) (Evrişimsel Sinir Ağları) kullanarak görüntü işleme yapılmıştır. CNN'ler, özellikle görsel verilerle çalışmak için tasarlanmış derin öğrenme modelleridir ve son yıllarda görüntü tanıma, nesne tespiti, yüz tanıma, otonom sürüş gibi birçok uygulamada büyük başarılar elde etmiştir. CNN'lerin temel avantajı, görsel verilerdeki mekansal bağımlılıkları öğrenebilme yetenekleridir. Görüntülerdeki pikseller arasındaki ilişkileri anlayarak, önemli özellikleri otomatik olarak çıkarabilirler.

CNN'ler, genellikle üç ana bileşenden oluşur:

1. **Evrişim Katmanları (Convolutional Layers):** Bu katmanlar, görüntüler üzerinde filtreler (kernels) uygulayarak özellik çıkarımı yapar. Her filtre, görüntüdeki belirli bir özelliği tanır (örneğin, kenarlar, köşeler, dokular) ve görüntü üzerinde bu özelliklerin yerlerini belirler. Bu aşama, görüntüdeki temel özelliklerin tanımlanmasına yardımcı olur.
2. **Havuzlama Katmanları (Pooling Layers):** Havuzlama, evrişim katmanlarının çıkardığı özelliklerin boyutlarını küçültmek için kullanılır. Bu katman, veriyi daha az yer kaplayacak şekilde özetlerken, önemli bilgileri kaybetmeden ağırlıkları azaltır. Max-pooling en yaygın kullanılan havuzlama yöntemidir ve görüntüdeki en yüksek değeri alarak boyutları küçültür.
3. **Tam Bağlantılı Katmanlar (Fully Connected Layers):** Bu katmanlar, daha önce çıkarılmış olan özellikleri sınıflandırma amacıyla işler. CNN'lerin son katmanında yer alan tam bağlantılı

katmanlar, görüntüdeki öğrenilen özellikleri kullanarak bir çıkış üretir. Bu çıkış, örneğin bir kırık olup olmadığını belirten bir sınıf etiketi olabilir.

CNN'ler, görüntülerin özelliklerini öğrenirken, her katmanda giderek daha soyut ve anlamlı temsiller oluşturur. İlk katmanlar basit özellikler (kenarlar, çizgiler) tanırken, derin katmanlar daha karmaşık yapıları (objeler, yüzeyler, nesneler) tanıyabilir. Bu özellik çıkarımı, görüntü işleme alanında büyük bir avantaj sağlar çünkü her katman, görüntüyü bir üst düzeye taşır. Bu sayede, model her bir görüntüyü daha anlamlı bir biçimde temsil edebilir ve doğru sınıflandırma yapabilir.

Görüntü işleme uygulamalarında CNN'ler, hem doğrusal (linear) hem de doğrusal olmayan (non-linear) özelliklerin öğrenilmesini sağlar. CNN'ler, görüntüdeki farklı nesnelerin ve yapıları tanıyabilme becerisi sayesinde, kırık tespiti gibi karmaşık görevlerde yüksek başarı elde eder. Bu çalışma da, X-ray kemik görüntülerinde derin öğrenme tabanlı nesne algılama (object detection) modelleri kullanarak, kemiklerdeki kırıkları doğru bir şekilde tespit etmeyi amaçlamaktadır.

CNN'ler, geleneksel yöntemlere göre birçok avantaja sahiptir. Özellikle, görüntülerdeki yüksek boyutluluk nedeniyle veri işleme açısından büyük bir avantaj sağlar. Bu ağlar, verinin mekansal ilişkilerini dikkate alarak görüntü üzerinde çok daha derinlemesine analizler yapabilir. Ayrıca, CNN'ler veri ile ilgili özellikleri otomatik olarak öğrenebildiği için, öznitelik mühendisliği (feature engineering) gibi manuel süreçlere olan ihtiyaç azalır.

Görüntü işleme problemlerinde, özellikle görüntüdeki kırıklar gibi küçük ve ince yapılar için CNN'ler etkili çözümler sunar. Bu modeller, verinin her bir bölgesini dikkatlice inceleyerek doğru ve yüksek doğrulukla tahminlerde bulunabilir. Örneğin, X-ray kemik görüntülerindeki kırıkların doğru bir şekilde tespit edilmesi için CNN'ler, kemik yapısındaki değişimleri tanıyabilir ve kırık olan bölgeleri işaret edebilir.

Her ne kadar CNN'ler görüntü işleme ve nesne algılama gibi alanlarda büyük başarılar elde etmiş olsa da, bu modellerin eğitiminde bazı zorluklar da bulunmaktadır. Özellikle, CNN'lerin eğitiminde büyük miktarda etiketli veri gereklidir ve modelin yüksek işlem gücüne sahip donanımlar ile eğitilmesi gerekir. Ayrıca, derin öğrenme modellerinin yorumlanabilirliği genellikle düşük olabilmektedir. Yani, CNN'lerin karar verme süreçleri genellikle "kara kutu" olarak kabul edilir ve bu durum, modelin nasıl sonuçlara ulaştığını anlamayı zorlaştırabilir.

Sonuç olarak, CNN'ler, derin öğrenme ile görüntü işleme alanında devrim yaratmış ve karmaşık verilerin analizinde büyük başarılar elde edilmesini sağlamıştır. Bu tez çalışmasında, derin öğrenme ve CNN kullanarak kemik kırıklarını doğru bir şekilde tespit etmeyi amaçlayan bir model geliştirilmiştir ve bu model, sağlık hizmetlerine önemli katkılar sağlayacaktır.

Deney

Modellerin Kurgulanacağı ve Çalıştırılacağı Ortamın Belirlenmesi:

Derin öğrenme modelleri, genellikle büyük veri kümeleri üzerinde çalıştığı için yüksek işlem gücü gerektirir. Bu nedenle, gelişmiş TPU (Tensor Processing Unit) ve GPU (Graphics Processing Unit) kaynakları, model eğitimi sürecinde kritik bir rol oynar. Ancak, kişisel bilgisayarlar çoğu zaman yeterli hesaplama kapasitesine sahip değildir. Bu noktada, Google Colab, ücretsiz veya Pro+ aboneliğiyle kullanıcılarına güçlü GPU kaynakları sunarak, model eğitimi için gereken hesaplama gücüne erişim sağlar.

Google Colab, bulut tabanlı bir platform olması nedeniyle, internet bağlantısı olan her cihazdan erişilebilir. Bu özellik, ekip çalışması ve projelerin farklı cihazlar arasında taşınabilmesi açısından büyük bir avantaj sağlar. Ayrıca, Colab defterleri kolayca paylaşılabilir ve geri bildirim almak, iş birliği yapmak oldukça basit hale gelir.

Google Colab, popüler derin öğrenme kütüphaneleri olan TensorFlow, PyTorch, Keras, Ultralytics gibi araçlarla tam uyumludur. Bu da, derin öğrenme modellerini geliştirirken kullanılan araçlarla kolayca entegrasyon sağlanmasını mümkün kılar. Bunun yanı sıra, Google Drive ile entegrasyon sunarak veri depolama ve iş akışı yönetimini basitleştirir.

Sonuç olarak, bu projede Google Colab kullanmak, yüksek işlem gücüne erişim sağlayarak verimli bir çalışma ortamı sunmaktadır. Bulut tabanlı çalışma imkânı sayesinde proje daha etkili bir şekilde gerçekleştirilebileceği için Google Colab tercih edilmiştir. Ayrıca projenin ilerleyen aşamalarında daha yüksek kapasiteli GPU'lara erişim sağlanması amacıyla Google Colab Pro+ aboneliği satın alınmıştır.

Veri Seti:

Derin öğrenme modelinin eğitimi ve testleri sırasında kullanılacak olan veri setinin oluşturulması, derin öğrenme modeli geliştirme uygulamaları için ilk ve en önemli adımdır. Projede kullanılan veri seti için, açık kaynak olarak paylaşılan "FracAtlas", "MURA (Musculoskeletal Radiographs)" gibi büyük veri setlerinden ve "Kaggle" internet sitesinde bulunan çeşitli kaynaklardan toplanarak, yaklaşık 250 adet kırık kemik görüntüsü elde edilmiştir. Veri setinden örnek bir görsel Şekil 1'de verilmektedir.

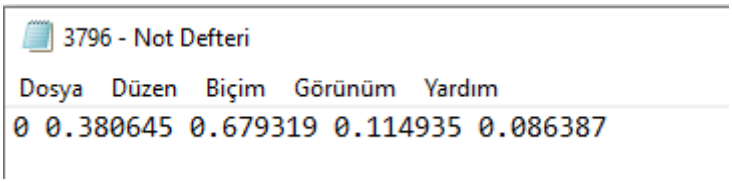
Şekil 1 Veri setinden örnek görsel

Elde edilen bu görüntüler, "Nesne Algılama" algoritmaları için gerekli olacak şekilde "etiketleme (labeling)" işlemlerine tabi tutulmuştur.

Burada bahsedilen etiketleme işlemi, derin öğrenme modelinin öğrenmesi ve tahmin etmesi gereken nesnelerin (proje için kırıklar) ilgili görüntü içerisindeki konumunun belirtildiği çeşitli dosyalar oluşturulması işlemidir ve sonrasında ilgili görüntüler ile etiket dosyalarının eşleştirilerek eğitim gerçekleştirilmesi için kritik önem taşımaktadır.

Etiketleme işlemleri, "makesense.ai" aracı kullanılarak gerçekleştirilmiştir. İlgili etiketleme işlemleri, derin öğrenme modellerinde ilk denemelerin yapıldığı YOLO formatında gerçeklenmiş olup, sonrasında diğer modellerin kullanımına uyum sağlayacak şekilde Python programlama dili kullanılarak dönüşümler gerçekleştirilmiştir. Şekil 1'deki örnek görselin etiketlenme işlemine ait görsel Şekil 2'de, oluşan etiket dosyasına ait görsel Şekil 3'te verilmiştir.

Şekil 2 makesense.ai ile etiketleme işlemi örneği



Şekil 3 YOLO formatında oluşan etiket dosyası

Burada oluşturulan etiket dosyası, YOLO formatında bir .txt dosyasıdır ve içerisinde 5 adet sayı barındırmaktadır. İlgili sayılar sırasıyla:

class x_center y_center width heigth

formatıyla ilerlemektedir.

class: İlgili görselin içerdiği veri sınıfını belirtir. Yapılan projede, yalnızca "Kırık (Fracture)" sınıfı olduğu için, başlangıç numarası olan "0" kullanılmıştır.

x_center, y_center: İlgili nesneye ait sınırlayıcı kutunun (bounding box) merkezinin, görüntünün yatay ve dikey eksenlerine göre normalize edilmiş değerleridir.

width: İlgili nesneye ait sınırlayıcı kutunun genişliğinin normalize edilmiş değeridir.

heigth: İlgili nesneye ait sınırlayıcı kutunun uzunluğunun normalize edilmiş değeridir.

Derin ağın daha genel öğrenebilmesi ve kemiklere özgü bazı durumların kırıklar ile karşılaştırılmaması için, veri setindeki görsellerde, kırık içeren X-Ray kemik görüntülerinin yanısıra, kırık içermeyen X-Ray kemik görüntüleri de kullanılmıştır. Kırık içermeyen X-Ray kemik görüntülerine ait .txt dosyalarının içleri boş bırakılmıştır. Böylece, oluşturulan derin; ağın eklem boşlukları, bilek boşlukları, parmak boşlukları gibi kısımları kırık olarak öğrenme durumları minimuma indirgenmeye çalışılmıştır.

Oluşturulan veri seti, derin ağ modeli oluşturma aşamasında, %70 eğitim, %15 validasyon (doğrulama) ve %15 test verisi olarak ayrıştırılmıştır.

Bu oran bazı çalışmalarda %60 - %20 - %20 olarak da ayarlanabilmektedir. Veri setindeki görüntülerin yanısıra görüntülere eşdeğer olan etiket dosyalarının da ayrıştırılması gerekmektedir. Bu işlemi yapan koda ait ekran görüntüsü Şekil 4'te verilmiştir.

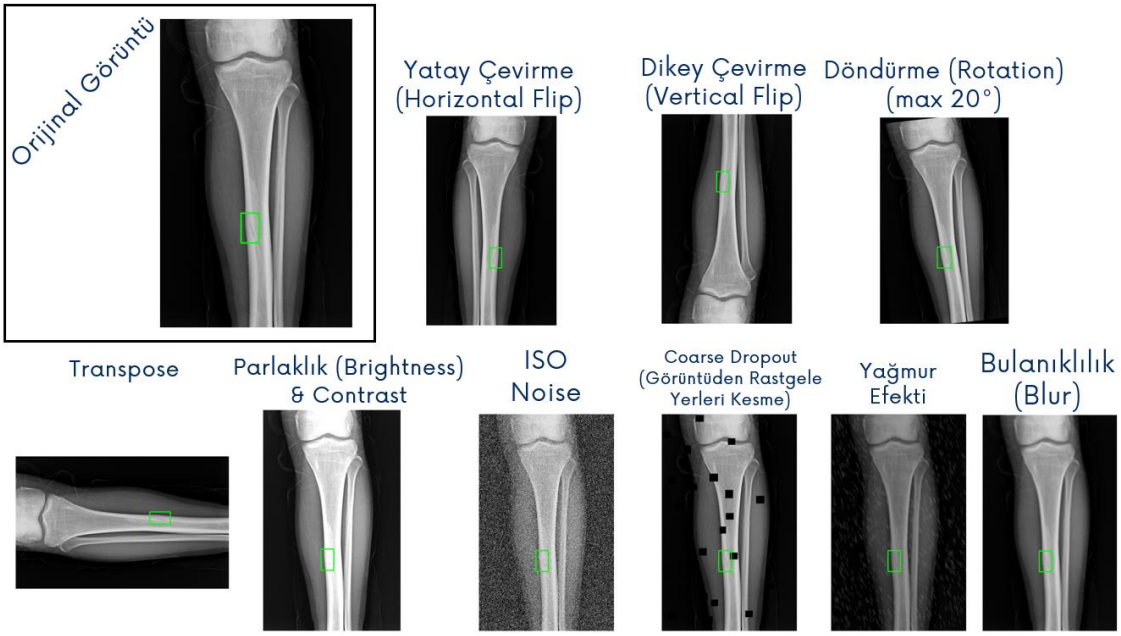
```
[ ] import os
    import shutil
    import random

    # Kaynak klasörler
    images_dir = "/content/tum/images"
    labels_dir = "/content/tum/labels"

    # Hedef klasörler
    output_dirs = {
        "train": {"images": "dataset/train/images", "labels": "dataset/train/labels"},
        "test": {"images": "dataset/test/images", "labels": "dataset/test/labels"},
        "valid": {"images": "dataset/valid/images", "labels": "dataset/valid/labels"},
    }

    # Hedef klasörleri oluştur
    for split in output_dirs.values():
        os.makedirs(split["images"], exist_ok=True)
        os.makedirs(split["labels"], exist_ok=True)

    # Tüm image dosyalarını al ve eşleştir
    image_files = [f for f in os.listdir(images_dir) if f.endswith(".jpg")]
    random.shuffle(image_files)  # Rastgele karıştırma

    num_total = len(image_files)
    num_train = int(num_total * 0.7)
    num_valid = int(num_total * 0.15)
    num_test = num_total - num_train - num_valid

    splits = {
        "train": image_files[:num_train],
        "valid": image_files[num_train:num_train + num_valid],
        "test": image_files[num_train + num_valid:]
    }

    # Dosyaları ilgili klasörlere taşı
    for split, files in splits.items():
        for file in files:
            img_src = os.path.join(images_dir, file)
            label_src = os.path.join(labels_dir, file.replace(".jpg", ".txt"))

            img_dst = os.path.join(output_dirs[split]["images"], file)
            label_dst = os.path.join(output_dirs[split]["labels"], file.replace(".jpg", ".txt"))

            shutil.move(img_src, img_dst)
            if os.path.exists(label_src):  # Label dosyasını taşı
                shutil.move(label_src, label_dst)
```

Şekil 4 Veri setinin Eğitim-Validasyon-Test şeklinde bölünmesini sağlayan kodun ekran görüntüsü

İlgili veri seti, veri sayısının yetersizliği dolayısı ile "Veri Artırımı (Data Augmentation)" yöntemleri ile, sentetik veri oluşturma suretiyle sayıca arttırılmıştır.

Veri artırımı işlemi için, Python'da bulunan "Albumentations" kütüphanesi kullanılmıştır.

OpenCV ve NumPy tabanlı olan Albumentations kütüphanesi; Pytorch ve TensorFlow ile kolayca entegre edilebilirliği, bünyesinde 60'dan fazla görsel veri artırımı metodunu barındırırması, ve en önemlisi, görüntülere veri artırımı uygulamasının yanı sıra nesne algılama algoritmaları için gereken etiket dosyalarına da ilgili veri artırma işlemlerini uygulaması ile öne çıkmaktadır. Uygulanan bu veri artırma işlemlerine örnekler Şekil 5'te verilmiştir.



Şekil 5 Uygulanan örnek veri artırma işlemleri

Veri setinde yapılan işlemler sonucunda, çeşitli senaryolar üzerinden eğitimler ve performans testleri gerçekleştirilmiştir. Bahsi geçen ilgili veri seti senaryoları Tablo 1'de verilmiştir.

Tablo 1 Kullanılan veri seti senaryoları

	Senaryo 1	Senaryo 2	Senaryo 3	Senaryo 4
Senaryo İsmi	No Augmentation - All Fractured	No Augmentation - Mixed	Augmentation - All Fractured	Augmentation - Mixed
Açıklama	Görüntülerde veri artırımı uygulanmamış ve bütün görüntüler kırık kemik içermektedir.	Görüntülerde veri artırımı uygulanmamış ve kırık – sağlam kemik görüntüleri karışık olarak kullanılmıştır.	Görüntülerde veri artırımı uygulanmış ve bütün görüntüler kırık kemik içermektedir.	Görüntülerde veri artırımı uygulanmış ve kırık – sağlam kemik görüntüleri karışık olarak kullanılmıştır.
Toplam Görüntü Sayısı	257	514	2268	5268

Performans

Derin öğrenme modellerinde, modellerin yaptığı tahminlerin sınıflandırıldığı "Confusion Matrix (Hata Matrisi), Tablo 2'de verilmiştir:

Tablo 2 Confusion Matrix (Hata Matrisi)

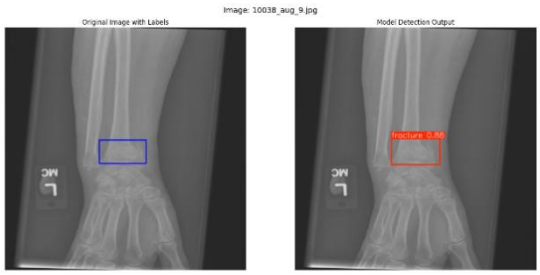
Prediction (Tahmin)	Positive (Pozitif)	True Positive (TP)	False Positive (FP)
	Negative (Negatif)	False Negative (FN)	True Negative (TN)
		Positive (Pozitif)	Negative (Negatif)
		Truth (Gerçek)	

İlgili durumların projedeki karşılıkları aşağıdaki gibidir:

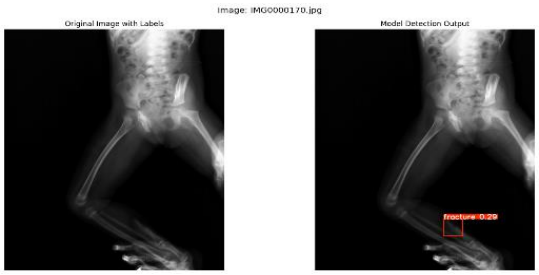
- TP (True Positive – Doğru Pozitif): Kırık bulunan bölgede kırık tespit edilmesi
- FP (False Positive – Yanlış Pozitif): Kırık bulunmayan bölgede kırık tespit edilmesi
- FN (False Negative – Yanlış Negatif): Kırık bulunan bir bölgede kırık tespit edilmemesi
- TN (True Negative – Doğru Negatif): Kırık bulunmayan bölgede kırık tespit edilmemesi

Hata matrisinde bahsedilen ilgili durumlara ait örnek görseller Şekil 6, 7, 8 ve 9'da verilmiştir.

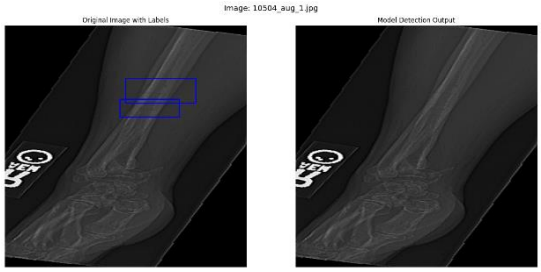
Şekillerde bulunan görüntülerde; Sol taraftaki "Mavi" kutucuk ile belirtilen görsel "Ground Truth (Gerçek Durum)", Sağ taraftaki "Kırmızı" kutucuk ile belirtilen görsel ise "Model Output (Model Çıktısı)"nı belirtmektedir.



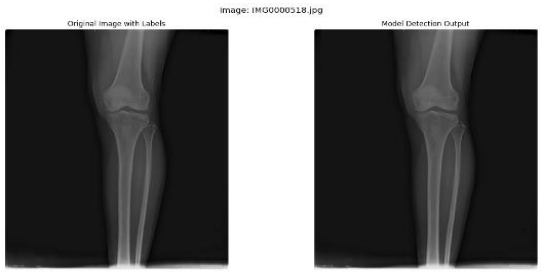
Şekil 6 True Positive (Doğru Pozitif)



Şekil 7 False Positive (Yanlış Pozitif)



Şekil 8 False Negative (Yanlış Negatif)



Şekil 9 True Negative (Doğru Negatif)

Bahsedilmesi gereken bir diğer önemli metrik ise “Intersection Over Union (IoU)”dur. IoU, nesne algılama algoritmalarının doğruluğunu değerlendiren önemli bir ölçüttür.

IoU, derin öğrenme modelinin tahmin ettiği sınırlayıcı kutunun (bounding box), gerçek sınırlayıcı kutu ile kesişim bölgesinin alanının, bu iki kutunun birleşim bölgesi alanına bölünmesi ile hesaplanır.

$$IoU = \frac{Kesişim\ Alanı\ (Intersection)}{Birleşim\ Alanı\ (Union)}$$

Projede eğitilen derin öğrenme modellerinin başarı durumları ve performansları karşılaştırılırken genel olarak dikkate alınan başarı metrikleri Tablo 3’deki gibidir.

Tablo 3 Başarı metriklerine ait açıklamalar

Metrik	**Tanım**	**Formül**	**Yorum**
Precision (Kesinlik)	Modelin yaptığı pozitif tahminlerin ne kadar doğru olduğunu gösterir. Yanlış pozitifleri azaltmaya odaklanır.	$= \frac{TP}{TP + FP}$	Yüksek olması, modelin yanlış pozitifler üretmediğini gösterir.
Recall (Duyarlılık)	Modelin tüm gerçek pozitifleri ne kadar iyi tespit ettiğini gösterir. Yanlış negatifleri azaltmaya odaklanır.	$= \frac{TP}{TP + FN}$	Yüksek olması, modelin tüm hedef nesneleri yakalayabildiğini gösterir.
mAP50 (Mean Average Precision at IoU 50%) **(IoU %50’de Ortalama Averaj Hassasiyet)**	Ortalama Precision değerinin, IoU (Intersection over Union) eşik değeri %50 olarak alındığında hesaplanmasıdır.	$= \frac{1}{N}\sum_{i=1}^{N} AP_i$	Modelin genel performansını ölçmek için kullanılır. Yüksek olması, modelin hem hassas hem de duyarlı olduğunu gösterir.

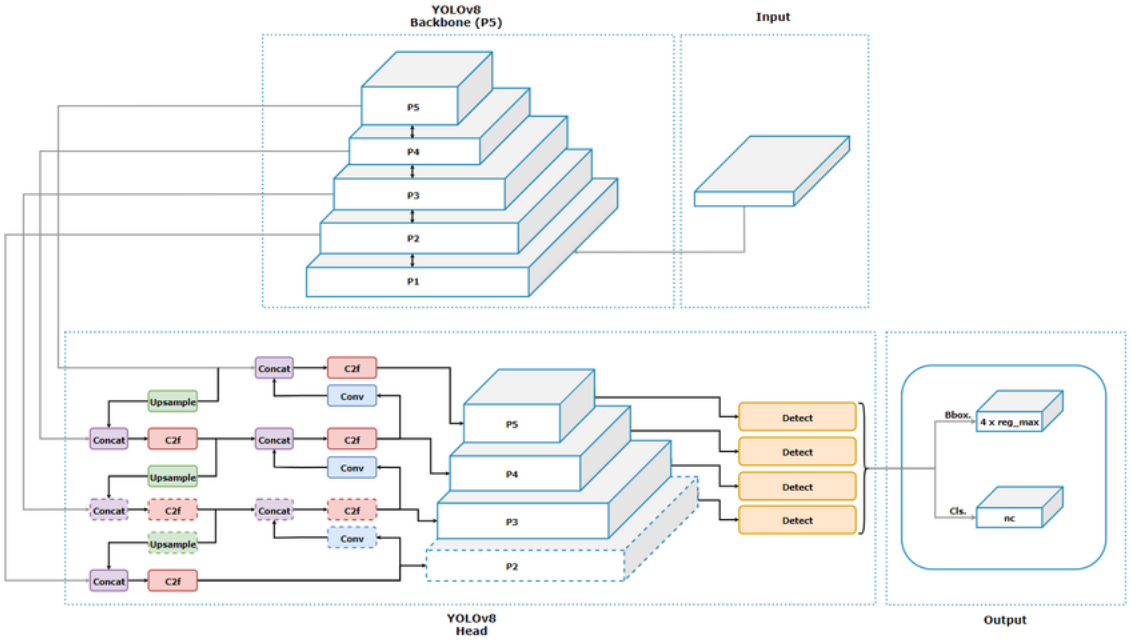
Ölçümler:

Yapılan literatür taraması sonrasında, nesne algılama algoritmalarından olan YOLO, RT-DETR, Faster R-CNN ve RetinaNet modelleri ile çalışılması kararı alınmıştır. Her bir modelin bir diğerinden farklı konuda üstünlüğü, yapılan çalışmayı çeşitlilik anlamında daha da zenginleştirmiştir.

YOLO (You Only Look Once):

YOLO, gerçek zamanlı nesne tespiti için geliştirilen tek aşamalı (one stage) bir model ailesidir. En güncel versiyonlardan olan YOLOv8, CSPDarknet temelli bir backbone ve C2f yapılı daraltılmış bloklarla donatılmıştır. (Şekil 10) Model, anchor-free ve decoupled head yaklaşımıyla çalışarak, sınıflandırma ve regresyon işlemlerini ayrı başlıklarda yürütür. Hem düşük gecikme süresi hem de

yüksek doğrulukla dikkat çeken YOLO mimarisi, özellikle küçük nesne tespitinde hızlı ve etkin bir çözüm sunar.



Şekil 8 YOLOv8 Model Mimarisi

Yapılan projede, ilk olarak nesne algılama algoritmalarının denenmesine YOLO ile başlanılmış ve YOLOv8 versiyonu tercih edilmiştir. YOLOv8'in alt versiyonları olan YOLOv8-S (Small – Küçük), YOLOv8-M (Medium – Orta), YOLOv8-L (Large – Büyük) ve YOLOv8-X (Extra Large – Ekstra Büyük) olacak şekilde pretrained (önceden eğitilmiş) modeller ile eğitimler gerçekleştirilmiştir. Modelin büyüklüğü, parametre sayısının fazlalığını belirtir, fakat model eğitiminin yavaşlaması gibi bir ters etki yaratmaktadır.

Yapılan eğitimler ve testlerde, YOLOv8-L versiyonu dışında, model büyütüldükçe performansta artış gözlemlenmiştir. YOLOv8-L modelinin öncülü olan YOLOv8-M modeli, üstün bir performansı daha kısa eğitim süresinde gösterebildiği için, YOLOv8-L modeli ile yapılan çalışma iptal edilmiştir.

YOLO modellerinin eğitimi için kullanılan kod bloğu Şekil 11'de verilmiştir. Model eğitimi için gereken fonksiyonların kullanımı amacıyla, Ultralytics kütüphanesi indirilmiş ve içerisinden YOLO fonksiyonları import edilmiştir. "model" değişkeni olarak, bahsedilen pretrained (önceden eğitilmiş) model indirme işlemi yapıldıktan sonra, "model.train" fonksiyonu içerisinde, kullanılacak olan hiperparametreler tanımlanmıştır:

```
[ ] !pip install ultralytics
    from ultralytics import YOLO

    model = YOLO('yolov8x.pt')

    # Train
    model.train(data = '/content/dataset_final/4_dataset_aug_mixed/data.yaml',
                epochs = 100,
                imgsz = 768,
                seed = 35,
                patience = 300,
                amp = False,


                lr0=0.001,    # Initial learning rate
                batch=16,     # Batch size
                freeze=10,    # Fine tuning
                warmup_epochs=5, # Fine tuning


                dropout=0.05, # Rastgele atlama
                iou=0.3, # Eğitim IoU eşiği
                optimizer="SGD"
    )
```

Şekil 9 YOLO Model Eğitim Kod Bloğu

data: Eğitimde kullanılacak veri setinin yolunu ve sınıf bilgilerini içeren ".yaml" dosyasıdır.

epochs=100: Modelin veri seti üzerinde 100 kez (epoch) eğitileceğini belirtir.

imgsz=768: Eğitimde kullanılacak giriş görüntülerinin boyutudur; 768x768 piksel olarak yeniden boyutlandırılır.

seed=35: Eğitim sürecindeki rastgeleliği sabitlemek için kullanılır.

patience=300: Early stopping için kullanılır; doğrulama kaybı (validation loss) 300 epoch boyunca iyileşmezse eğitim durur. Bu aşamada durdurulmaması için yüksek seçilmiştir.

amp=False: Mixed precision (16-bit) eğitim devre dışı bırakılmıştır; sadece 32-bit ile çalışılır.

lr0=0.001: Modelin ağırlıklarının ne kadar değişeceğini belirleyen öğrenme oranıdır; daha düşük değer, daha yavaş ama istikrarlı öğrenme sağlar.

batch=16: Her adımda modele verilecek görüntü sayısıdır; bellek tüketimi ve genel öğrenme dengesini etkiler.

freeze=10: Modelin ilk 10 katmanı eğitime kapatılır, sadece kalan katmanlar güncellenir; transfer learning sırasında kullanılır.

warmup_epochs=5: İlk 5 epoch boyunca öğrenme oranı yavaşça artırılır; ani ağırlık değişimlerini önler.

dropout=0.05: Eğitim sırasında %5 oranında rastgele nöron devre dışı bırakılarak aşırı öğrenmenin (overfitting) önüne geçilir.

iou=0.3: Eğitim sırasında kullanılan Intersection over Union eşiğidir; tahmin kutusunun doğru kabul edilmesi için gerekli minimum örtüşme oranıdır.

optimizer="SGD": Ağırlık güncellemesinde kullanılan algoritmadır; Stochastic Gradient Descent klasik ve kontrollü bir optimizasyon tekniğidir.

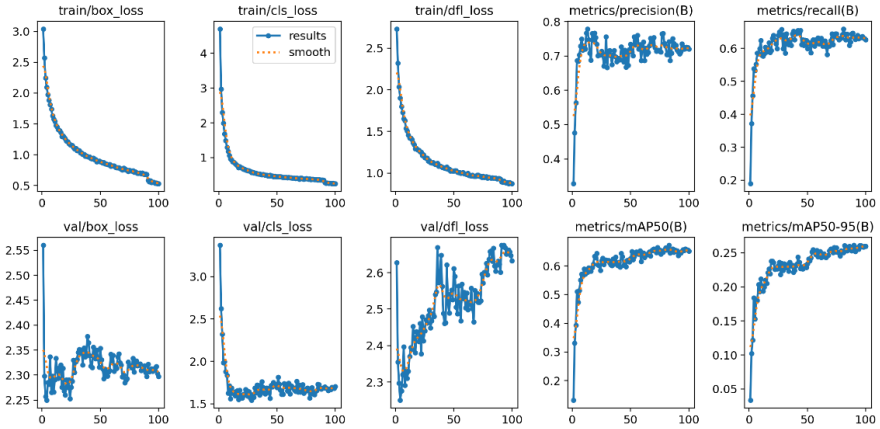
Kullanılan YOLOv8 alt versiyonları ile, Tablo 1'de bahsedilen farklı veri seti senaryoları denenerek elde edilen test sonuçları Tablo 4'te verilmiştir.

Tablo 4 YOLOv8 Test Sonuçları

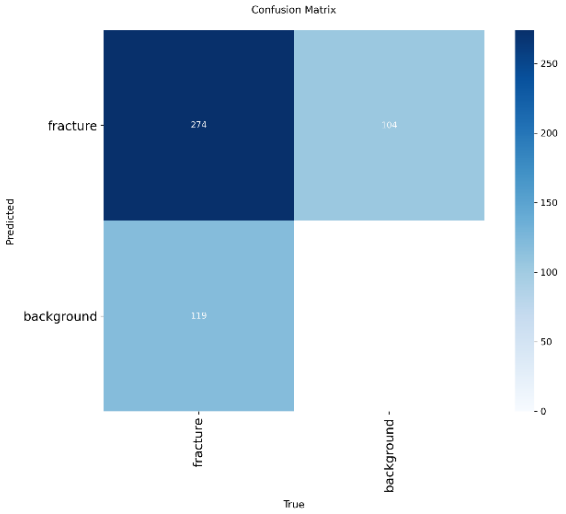
	Senaryo 1			Senaryo 2			Senaryo 3			Senaryo 4		
	No Augmentation - All Fractured			No Augmentation - Mixed			Augmentation - All Fractured			Augmentation - Mixed		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
YOLOv8-S	79.5%	47%	61.3%	76.2%	46.5%	55.8%	74.9%	45.9%	61.2%	83%	44.1%	%65.5
YOLOv8-M	69.7%	60.9%	63.8%	68.9%	67.4%	70.6%	74.7%	64.1%	69%	74.1%	65.6%	%71.2
YOLOv8-X	84.4%	58.7%	70.2%	65.2%	65.2%	67.9%	73%	65.1%	71.8%	78%	64.1%	%71.5

Denenen farklı veri seti senaryoları ve farklı model büyüklükleri karşılaştırıldığında, ana başarı metriği olan mAP50 özelinde en başarılı modelin "Senaryo 3 – YOLOv8-X" olduğu gözlenmiştir.

İlgili modele ait eğitim grafikleri Şekil 12'de, hata matrisi grafiği Şekil 13'de verilmiştir.



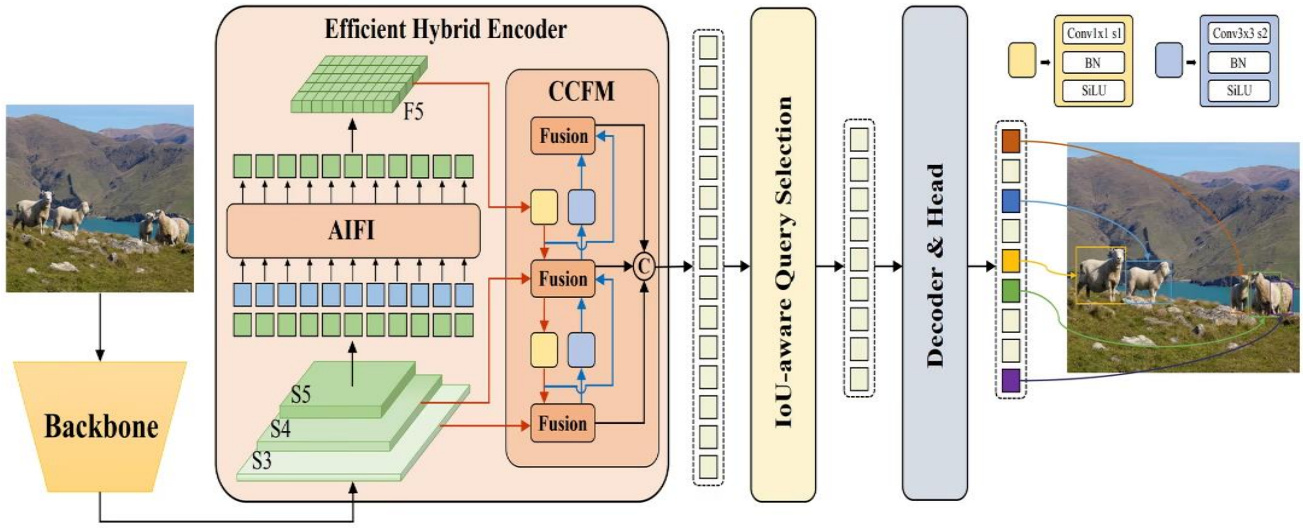
Şekil 10 En başarılı YOLO modelinin Eğitim - Validasyon Grafikleri



Şekil 11 En başarılı YOLO modelinin Test Hata Matrisi

RT-DETR (Real-Time Detection Transformer)

RT-DETR, detection transformer mimarisinin verimli hale getirilmiş bir versiyonudur ve tamamen transformer tabanlı bir yaklaşıma sahiptir. Model, backbone olarak genellikle ResNet veya Swin Transformer kullanırken, sorgu tabanlı (query-based) bir dekoder yapısıyla anchor box veya NMS (non-max suppression) kullanmadan doğrudan sınırlayıcı kutular üretir. (Şekil 14) Denoising training gibi yöntemlerle eğitimi hızlandırılmış olan RT-DETR, bağlam ilişkilerini güçlü şekilde modelleyerek küçük nesne tespitinde derin uzamsal farkındalık sunar.



Şekil 12 RT-DETR Model Mimarisi

RT-DETR eğitimi için, iki farklı versiyonu olan RT-DETR-L (Large – Büyük) ve RT-DETR-X (Extra Large – Ekstra Büyük) olacak şekilde pretrained (önceden eğitilmiş) modeller ile eğitimler gerçekleştirilmiştir. Modelin büyüklüğü, parametre sayısının fazlalığını belirtir, fakat model eğitiminin yavaşlaması gibi bir ters etki yaratmaktadır.

RT-DETR derin ağ modeli, YOLO ile benzer şekilde Ultralytics tarafından geliştirilmiştir ve benzer fonksiyonları kullanır. Hiperparametre ayarlamaları, Şekil 11'de belirtildiği şekilde, YOLO modellerinin eğitimi ile birebir aynı gerçekleştirilmiştir.

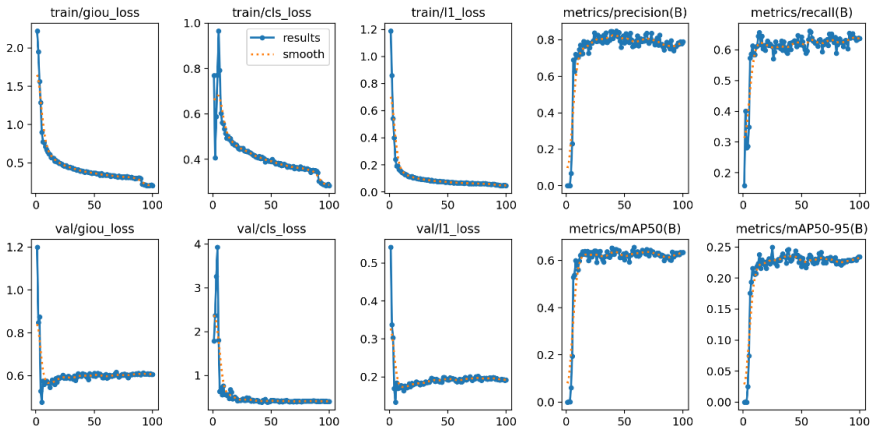
Kullanılan RT-DETR alt versiyonları ile, Tablo 1'de bahsedilen farklı veri seti senaryoları denenerek elde edilen test sonuçları Tablo 5'te verilmiştir.

Tablo 5 RT-DETR Test Sonuçları

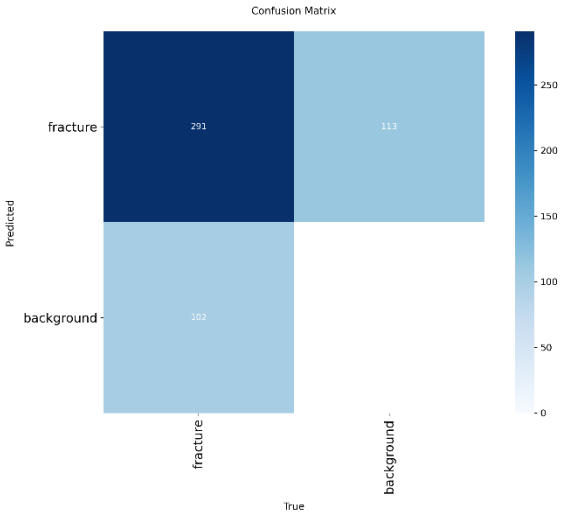
	Senaryo 1			Senaryo 2			Senaryo 3			Senaryo 4		
	No Augmentation - All Fractured			No Augmentation - Mixed			Augmentation - All Fractured			Augmentation - Mixed		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
RT-DETR-L	25.8%	18.3%	18.3%	43.9%	26.1%	35.4%	72.8%	53.2%	62%	71.3%	63.6%	68%
RT-DETR-X	66.1%	60.9%	68.7%	60%	58.7%	56.1%	72%	63.9%	68.6%	75.6%	65.9%	72.9%

Denenen farklı veri seti senaryoları ve farklı model büyüklükleri karşılaştırıldığında, ana başarı metriği olan mAP50 özelinde en başarılı modelin “Senaryo 4 – RT-DETR-X” olduğu gözlenmiştir.

İlgili modele ait eğitim grafikleri Şekil 15’te, hata matrisi grafiği Şekil 16’da verilmiştir.



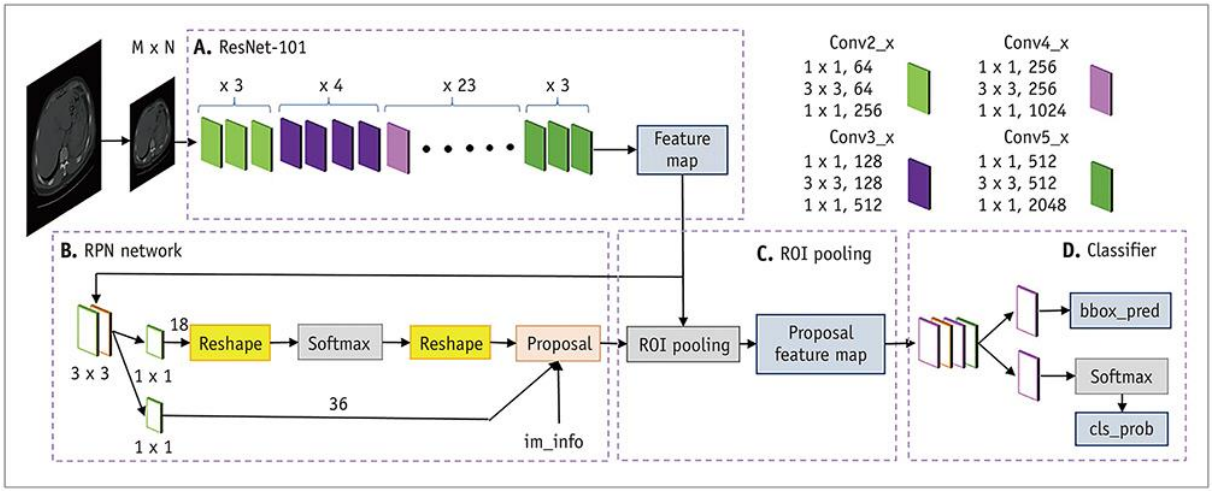
Şekil 13 En başarılı RT-DETR modelinin Eğitim - Validasyon Grafikleri



Şekil 14 En başarılı RT-DETR modelinin Test Hata Matrisi

Faster R-CNN

Faster R-CNN, iki aşamalı (two stage) bir nesne tespit modelidir ve Region Proposal Network (RPN) ile aday bölgeler ürettikten sonra bu bölgeleri sınıflandırarak son karar verir. Backbone olarak genellikle ResNet + FPN (Feature Pyramid Network) kombinasyonu kullanılır (Şekil 17). Mimarisindeki bu ayrık aşamalar sayesinde doğruluğu yüksek olmasına rağmen, işlem süresi açısından daha yavaştır. Anchor tabanlı yapı ve RPN'nin hassas ayar gereksinimi, modelin küçük nesne tespiti gibi görevlerde dikkatli konfigürasyon gerektirmesine yol açar.



Şekil 15 Faster R-CNN Model Mimarisi

Faster R-CNN modelinin eğitiminde, YOLO ve RT-DETR'dan farklı olarak "Detectron2" kütüphanesinden yararlanılmıştır. Detectron2, Facebook AI Research (FAIR) tarafından geliştirilen, PyTorch tabanlı gelişmiş bir nesne tespiti kütüphanesidir.

İlk olarak, YOLO formatındaki etiket dosyalarının Detectron2 kütüphanesinin kullandığı COCO JSON formatına uygun hale getirilmesi gerekmektedir. Bu işlem için bir Python scripti yazılmıştır.

Bu işlem yapıldıktan sonra, Detectron2 konfigürasyon dosyası oluşturulup eğitimde kullanılacak olan hiperparametreler belirlenmiştir. İlgili konfigürasyon dosyasına ait ekran görüntüsü Şekil 18'de verilmiştir.

```
# Model Configuration
cfg = get_cfg()

# Output Directory
Path(Config.PATHS["output"]).mkdir(parents=True, exist_ok=True)

try:
    cfg.merge_from_file(model_zoo.get_config_file(Config.MODEL))
except RuntimeError as e:
    print(f"Error loading model: {e}")

# Model Hyperparameters
cfg.DATASETS.TRAIN = ("fracture_train",)
cfg.DATASETS.TEST = ("fracture_test",)
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(Config.MODEL)
cfg.SOLVER.IMS_PER_BATCH = 16  # Batch size, 8-16
cfg.SOLVER.BASE_LR = 0.001 # Learning rate
cfg.SOLVER.MAX_ITER = 2275    # Max iterasyon = (Train görüntü sayısı * Epoch) / (Batch size)
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64  # 32-64 sistem gereksinimini azaltır ama performans düşer, 96-128-256 sistem gereksinimi artar performans artar
cfg.MODEL.ROI_HEADS.NUM_CLASSES = len(Config.CLASSES)
cfg.SOLVER.CHECKPOINT_PERIOD = 2276


cfg.OUTPUT_DIR = Config.PATHS["output"]
```

Şekil 16 Faster R-CNN Detectron2 Konfigürasyon Dosyası

cfg.DATASETS.TRAIN & TEST: Eğitim ve test veri setlerinin adını belirler.

cfg.DATALOADER.NUM_WORKERS: Veri yükleyici işlemlerinde kullanılacak CPU iş parçacığı (thread) sayısıdır.

cfg.MODEL.WEIGHTS: Eğitimi başlatmak için kullanılacak olan pretrained (önceden eğitilmiş) model ağırlıklarının yoludur.

cfg.SOLVER.IMS_PER_BATCH: Her iterasyonda modele verilecek görüntü sayısıdır (batch size).

cfg.SOLVER.BASE_LR: Modelin öğrenme oranını belirler, ağırlıkların ne kadar güncelleneceğini kontrol eder.

cfg.SOLVER.MAX_ITER: Toplam eğitim iterasyon (adım) sayısını belirler. YOLO ve RT-DETR'da kullanılan Epoch kavramı ile arasındaki ilişki aşağıdaki gibidir:

$$Maksimum\ Iterasyon = \frac{Eğitim\ Görüntü\ Sayısı * Epoch\ Sayısı}{Batch\ Size}$$

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE: Her görüntüde ROI başlıklarına gönderilecek örnek sayısıdır; düşük değer sistem ihtiyacını azaltır, yüksek değer doğruluğu artırır.

cfg.MODEL.ROI_HEADS.NUM_CLASSES: Modelin kaç farklı sınıf tespiti yapacağını belirtir.

cfg.SOLVER.CHECKPOINT_PERIOD: Kaç iterasyonda bir modelin checkpoint olarak kaydedileceğini belirler. Bu değer ayarlanmazsa otomatik 5000 iterasyonda bir model kaydı yapılır.

cfg.OUTPUT_DIR: Eğitim çıktı dosyalarının (log, model vb.) kaydedileceği dizini belirtir.

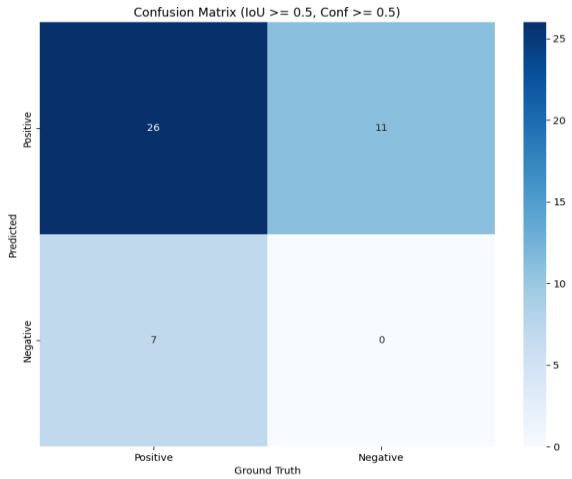
Kullanılan Faster R-CNN konfigürasyonu ile, Tablo 1'de bahsedilen farklı veri seti senaryoları denenerek elde edilen test sonuçları Tablo 6'da verilmiştir.

Tablo 6 Faster R-CNN Test Sonuçları

	Senaryo 1			Senaryo 2			Senaryo 3			Senaryo 4		
	No Augmentation - All Fractured			No Augmentation - Mixed			Augmentation - All Fractured			Augmentation - Mixed		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
Faster R-CNN	56.89%	64.7%	56.62%	70.27%	78.79%	69.98%	55.35%	67.15%	57.08%	62.8%	68.42%	58.9%

Denenen farklı veri seti senaryoları karşılaştırıldığında, ana başarı metriği olan mAP50 özelinde en başarılı modelin "Senaryo 2" olduğu gözlenmiştir.

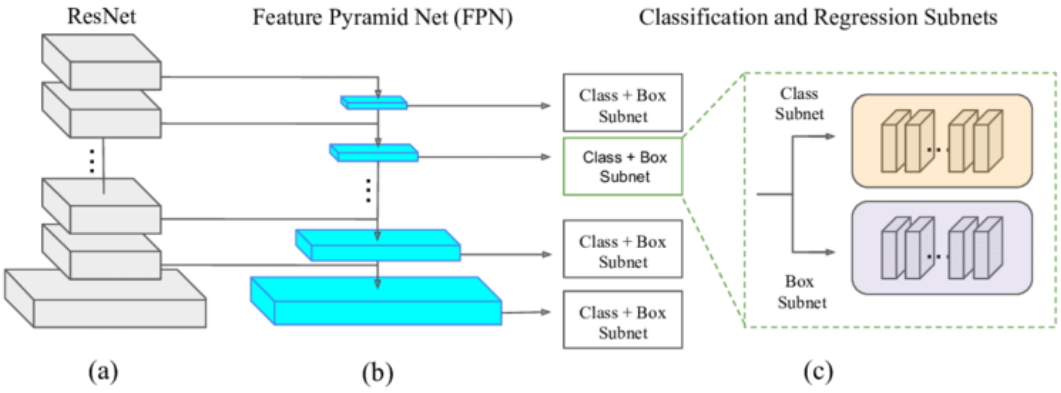
İlgili modele ait hata matrisi grafiği Şekil 19'da verilmiştir.



Şekil 17 En başarılı Faster R-CNN modelinin Test Hata Matrisi

RetinaNet

RetinaNet, tek aşamalı (one stage) bir nesne algılama modelidir ve özellikle Focal Loss fonksiyonu sayesinde sınıf dengesizliği sorununu etkin biçimde çözer. Backbone olarak ResNet ve FPN kombinasyonunu kullanan RetinaNet, farklı ölçeklerdeki nesneleri algılayabilmek için çok seviyeli özellik haritalarını değerlendirir. (Şekil 20) Basit ve etkili mimarisi sayesinde orta seviyede doğruluk ve hız sunar; ancak veri setinde negatif örneklerin oranı arttığında yanlış pozitif üretme eğiliminde olabilir.



Şekil 18 RetinaNet Model Mimarisi

RetinaNet modelinin eğitiminde, önceki üç modelden farklı olarak "Torchvision" kütüphanesinden yararlanılmıştır. Torchvision, PyTorch ekosistemi içinde yer alan ve hazır modeller, veri setleri, dönüşüm araçları sağlayan bir kütüphanedir.

İlk olarak, YOLO formatındaki etiket dosyalarının Torchvision kütüphanesinin kullandığı CSV formatına uygun hale getirilmesi gerekmektedir. Bu işlem için bir Python scripti yazılmıştır.

RetinaNet modellerinin eğitimi için kullanılan kod bloğu Şekil 21'de verilmiştir.

```
[ ] from torchvision.models.detection import retinanet_resnet50_fpn

    device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
    model = retinanet_resnet50_fpn(pretrained=False, num_classes=1)
    model.to(device)
    model.train()
    optimizer = torch.optim.SGD([p for p in model.parameters() if p.requires_grad], lr=0.001, momentum=0.9, weight_decay=0.0005)

    for epoch in range(100):
        for images, targets in data_loader:
            images = [img.to(device) for img in images]
            targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

            loss_dict = model(images, targets)
            losses = sum(loss for loss in loss_dict.values())

            optimizer.zero_grad()
            losses.backward()
            optimizer.step()

        print(f"Epoch {epoch+1} Loss: {losses.item():.4f}")

    torch.save(model.state_dict(), 'retinanet_weights.pth')
```

Şekil 19 RetinaNet Model Eğitim Kod Bloğu

lr=0.001: Modelin ağırlıklarının ne kadar değişeceğini belirleyen öğrenme oranıdır; daha düşük değer, daha yavaş ama istikrarlı öğrenme sağlar.

momentum=0.9: SGD optimizasyonunda, önceki adımların yönünü koruyarak öğrenmeyi hızlandıran ivme katsayısıdır.

weight_decay=0.0005: Eğitim sırasında %0.5 oranında rastgele nöron devre dışı bırakılarak aşırı öğrenmenin (overfitting) önüne geçilir.

num_classes=1: Eğitimde kullanılacak hedef sınıf sayısıdır. (Kırık)

epochs=100 (for satırı içinde range(100)): Modelin veri seti üzerinde 100 kez (epoch) eğitileceğini belirtir.

optimizer="SGD": Ağırlık güncellemesinde kullanılan algoritmadır; Stochastic Gradient Descent klasik ve kontrollü bir optimizasyon tekniğidir.

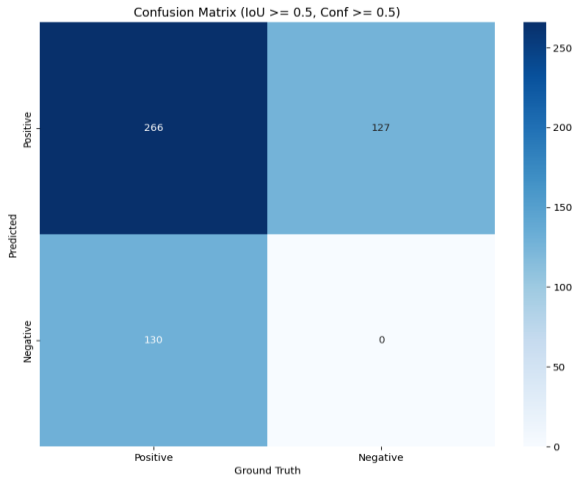
Kullanılan RetinaNet konfigürasyonu ile, Tablo 1'de bahsedilen farklı veri seti senaryoları denenerek elde edilen test sonuçları Tablo 7'de verilmiştir.

Tablo 7 RetinaNet Test Sonuçları

	Senaryo 1			Senaryo 2			Senaryo 3			Senaryo 4		
	No Augmentation - All Fractured			No Augmentation - Mixed			Augmentation - All Fractured			Augmentation - Mixed		
	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50	Precision	Recall	mAP50
RetinaNet	59.46%	47.83%	59.46%	43.08%	60.87%	43.08%	67.17%	67.68	67.17%	47.03%	64.38%	47.03%

Denenen farklı veri seti senaryoları karşılaştırıldığında, ana başarı metriği olan mAP50 özelinde en başarılı modelin "Senaryo 3" olduğu gözlenmiştir.

İlgili modele ait hata matrisi grafiği Şekil 22'de verilmiştir.



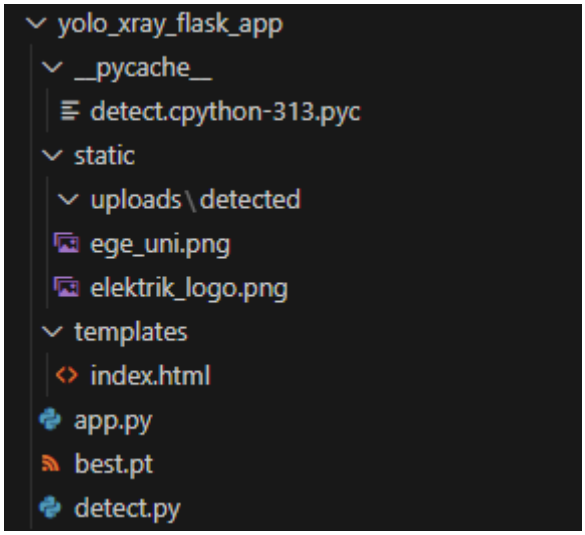
Şekil 20 En başarılı RetinaNet modelinin Test Hata Matrisi

4.SONUÇLAR

Denenen farklı veri seti senaryoları ve farklı modeller karşılaştırıldığında, ana başarı metriği olan mAP50 özelinde, proje içerisinde en başarılı modelin "Senaryo 4 – RT-DETR-X" olduğu gözlenmiştir.

Projede, eğitilmiş RT-DETR nesne algılama modelini son kullanıcıya ulaştırmak amacıyla Python tabanlı Flask mikro web çatısı kullanılarak bir arayüz geliştirilmiştir. Bu arayüz, kullanıcıların kendi X-Ray görüntülerini yükleyerek model çıktısını doğrudan görsel olarak alabilecekleri kullanıcı dostu bir sistem sunar.

Geliştirilen sistem; görsel yükleme, modelin çalıştırılması ve elde edilen sonucun web arayüzü üzerinden kullanıcıya sunulması gibi adımları bütünleşik bir yapıda birleştirmektedir. Ayrıca, sade bir tasarım anlayışı ile birlikte sayfa geçişleri ve etkileşimlerde kullanılan animasyonlar sayesinde kullanıcı deneyimi artırılmıştır. Bu bölümde geliştirilen sistemin üç temel bileşeni olan detect.py, app.py ve HTML dosyasına ait açıklamalar, ekran görüntüleri ile birlikte sunulmaktadır.



Şekil 21 Arayüz Proje Dosyası ve Klasör Yolları

```python
import os
import cv2
from ultralytics import RTDETR

model = RTDETR("best.pt")  # Model dosyası
def run_detection(image_path):
    results = model(image_path)
    result = results[0]

    img = cv2.imread(image_path)
    for box in result.boxes.xyxy:
        x1, y1, x2, y2 = map(int, box)
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

    output_path = image_path.replace("uploads", "uploads/detected")
    os.makedirs(os.path.dirname(output_path), exist_ok=True)
    cv2.imwrite(output_path, img)
    return output_path
```

Şekil 22 Model Tespit Kodu (detect.py)

detect.py dosyası, (Şekil 24) RT-DETR modelinin yüklenmesi ve bir görüntü üzerinde nesne tespiti yapılması işlemlerini barındırır. Kullanıcı tarafından yüklenen X-Ray görüntüsü bu dosyadaki run_detection() fonksiyonu aracılığıyla işlenir. Fonksiyonun yaptığı işlemler:

- best.pt adlı model dosyası yüklenir.
- Görüntü, RT-DETR modeliyle analiz edilir.
- Model çıktısındaki bounding box koordinatları alınarak görüntü üzerine yeşil kutular çizilir.
- Elde edilen sonuç görüntüsü uploads/detected/ klasörüne kaydedilir.
- Kaydedilen çıktı dosyasının yolu döndürülerek arayüzde kullanılmak üzere Flask tarafına iletilir.

Bu yapı sayesinde, görsel işleme tamamen model tarafında otomatik olarak gerçekleştirilmekte ve kullanıcıya doğrudan sonuç sunulmaktadır.

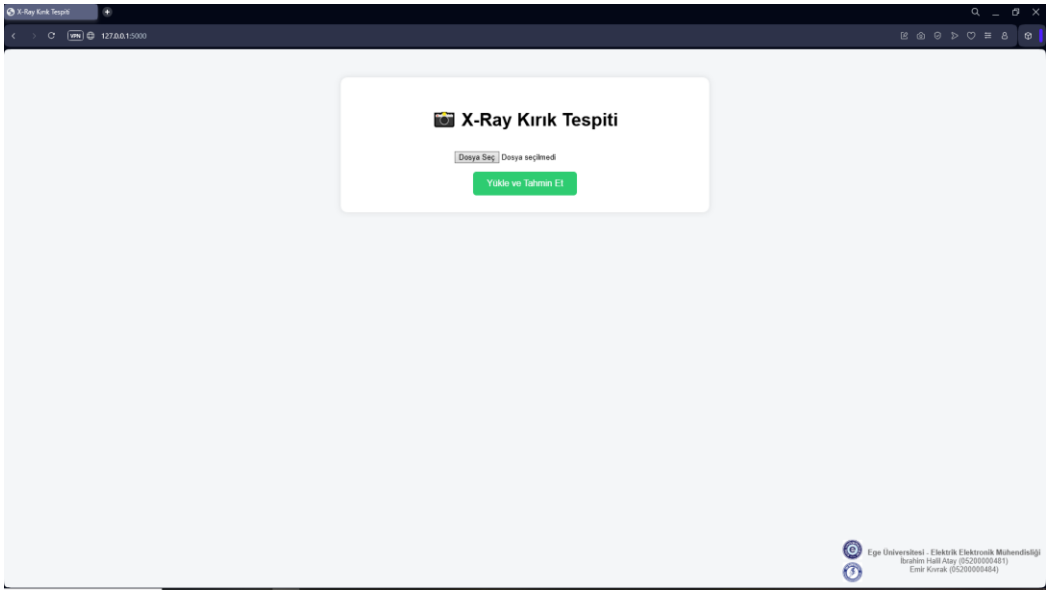
app.py, Flask uygulamasının merkezini oluşturur. (Şekil 25) Kullanıcıdan alınan POST isteklerini işler, yüklenen dosyaları sunucuya kaydeder, detect.py üzerinden model çalıştırır ve sonucu HTML arayüzüne yönlendirir.

- Flask framework'ü başlatılır ve "static/uploads" klasörü hazırlanır.
- Ana sayfa ("/") yönlendirmesi yapılır, hem GET (sayfa açılışı) hem POST (görüntü yükleme) istekleri desteklenir.
- Kullanıcı bir dosya yüklediğinde, dosya "zaman etiketli" özel bir adla kaydedilir. Bu sayede dosyaların karışmasının önüne geçilir.
- Kaydedilen dosya model fonksiyonuna gönderilir ve çıktı alınır.
- Sonuç, index.html üzerinden kullanıcıya gösterilir.
- Sayfa yeniden yüklendiğinde form sıfırlanır ve yeni bir görsel yüklenmesine izin verilir.

Bu yapı, modelin arayüzle etkileşimini yönetir ve tüm iş akışını koordine eder.

```python
from flask import Flask, render_template, request
import os
from detect import run_detection
from datetime import datetime

app = Flask(__name__)
UPLOAD_FOLDER = "static/uploads"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        image = request.files["xray"]
        if image:
            filename = datetime.now().strftime("%Y%m%d%H%M%S") + "_" + image.filename
            filepath = os.path.join(UPLOAD_FOLDER, filename)
            image.save(filepath)

            # Tahmin çalıştır
            output_path = run_detection(filepath)

            return render_template("index.html", uploaded=True, output_image=output_path)

    return render_template("index.html", uploaded=False)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)

```

Şekil 23 Uygulama Kodu (app.py)



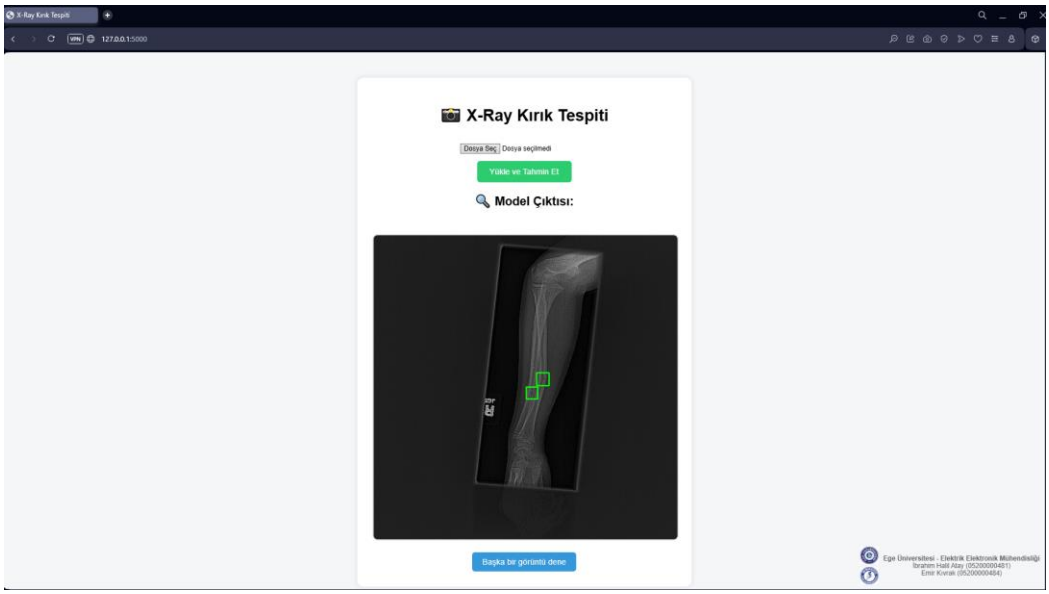
Şekil 24 Arayüz Ana Sayfası

templates/index.html/ dosya yolunda bulunan HTML dosyası, web tarayıcısı üzerinden kullanıcıyla etkileşim kuran arayüzü tanımlar. (Şekil 26) Görsel yükleme, sonuçların görüntülenmesi ve kullanıcı yönlendirmeleri bu dosya ile sağlanır. Yapının özellikleri:

- form bloğu ile kullanıcıdan .jpg, .png gibi görüntü dosyaları alınır.
- Görsel yüklendiğinde, sonucu gösteren img etiketi devreye girer.
- Sayfa genelinde fade-in animasyonlar kullanılarak içerikler yumuşak geçişlerle ekrana gelir.

- Sağ alt köşeye sabitlenmiş bir footer bileşeni ile proje sahiplerinin isimleri ve öğrenci numaraları gösterilir.
- Aynı alanda Ege Üniversitesi ve Elektrik Elektronik Mühendisliği logoları alt alta yer alır.

Bu HTML arayüzü, hem kullanıcı deneyimini iyileştirmekte hem de proje kimliğini estetik şekilde yansıtmaktadır.



Şekil 25 Arayüz Model Tespiti Örneği

Tartışmalar ve Öneriler

Elde edilen sonuçlarda, hedef olan %70+ mAP@50 (Mean Average Precision at IoU %50) oranına ulaşılsa da, bu oran genel anlamda nesne algılama uygulamaları için düşük bir orandır. Bunun sebebi, nesne algılama modellerinin günümüz şartlarında küçük nesneler için yeteri kadar iyi çalışamıyor olmasıdır. Nesne algılama modellerinin üzerine yapılacak yeni çalışmalar ile küçük nesneler için optimize edilmeleri, kırık ve kırık gibi küçük nesnelere yönelik çalışmalardaki başarının artması için önemli bir adım olacaktır.

Ayrıca günümüz şartlarında, nesne algılama uygulamaları arasında bir numara olan YOLO ve aynı yapımcı tarafından geliştirilen RT-DETR üzerine çok kaynak bulunsa da, projede kullanılan diğer modeller olan Faster R-CNN ve RetinaNet üzerine kaynak yetersizdir. Özellikle RetinaNet gibi "Focal Loss" ile küçük nesnelere karşı avantaj sağlayan modeller üzerine daha fazla çalışılması gerekmektedir. YOLO ve RT-DETR genel kapsamlı çalışmalar için daha hızlı ve optimize çalışmaları ile öne çıkmaktadır.

Fazla sayıda verinin çeşitli şekillerde incelenmesi durumunda, basit modeller yerine daha karmaşık ve derin modellerin kullanılması gerekmektedir. Bu durum, modelin eğitim sürecinde daha fazla kaynak ve zaman harcanmasına yol açsa da Google Colab Pro+ gibi servisler sayesinde, yüksek kapasiteli ve güçlü GPU'lar ile bu problemin üstesinden gelinebilmektedir.

Yetersiz veri miktarıyla yapılan eğitim ve test süreçlerinde, doğruluk oranlarında tutarsızlıklar yaşanması olasılığı bulunmaktadır. Bu sorunu aşabilmek için, test veri setinin boyutunun, eğitim veri setinin en az %30'u kadar olması gerekmektedir. Ayrıca, veri setinin eğitim, test ve doğrulama (validasyon) olarak üçe ayrılması ve bu verilerin modele iteratif şekilde verilmesi, modelin eğitim verilerini ezberlemesini engelleyecektir.

İleriye Dönük Çalışmalar

Gelecekteki araştırmalar ve gelişim süreçlerinde, modellerin tahmin yeteneğini arttırmak amacıyla, model için kullanılacak görüntülerin kalitesinin yükseltilmesi birinci öncelik olmalıdır. Günümüz şartlarında X-Ray kemik görüntüleri üzerine kaliteli açık kaynak veri bulmak zordur.

Modelin daha geniş bir kullanıcı kitlesine hitap edebilmesi için, farklı yaş grupları, cinsiyetler ve kemik yapıları gibi çeşitli demografik özelliklere sahip daha kapsamlı veri setleri oluşturulmalıdır. Ayrıca, eğitilen model için, kullanılan veri setindeki kemiklerin vücut bölgesine göre eşitlenmesi ve modelin her bölgeyi eşit şekilde öğrenmesi kritik önem sarfetmektedir. Buna ek olarak, vücut bölgeleri veya kırık tipleri ayrı birer sınıf olarak kullanılıp model eğitimi farklı yönde evrilebilir. Model, yeni verilerle sürekli güncellenip yeniden eğitilerek sürekliliği korunmalıdır.

Ayrıca, kullanıcıların ve sağlık çalışanlarının, X-Ray görüntülerindeki kırıkları daha doğru bir şekilde tespit edebilmeleri için eğitim materyalleri ve rehberler sağlanmalıdır. Klinik ortamda uygulanabilirliğin test edilmesi adına, sağlık kurumlarıyla iş birliği yapılmalı ve modelin gerçek dünya koşullarındaki performansı detaylı bir şekilde incelenmelidir.

İncelenen modeller arasında özellikle RetinaNet modeli, “Focal Loss” ile küçük nesnelere karşı sağladığı avantaj ile öne çıkmaktadır. Yapılan çalışmalarda, başarı oranı nispeten diğer denenen modellere göre daha düşük olsa da, ilgili konuda üzerine çalışılması önerilen bir derin ağ modeli olmuştur.

Teşekkür

Proje ekibi olarak, tez çalışmasını beraber yürüttüğümüz, süreç boyunca yol göstericimiz olan ve bizlere desteğini eksik etmeyen çok değerli danışman hocamız Doç. Dr. Erkan Zeki Engin’e ve 2209-A Üniversite Öğrencileri Araştırma Projeleri Destekleme Programı kapsamında projemizi destekleyen TÜBİTAK’a en içten teşekkürlerimizi sunarız.

Kaynakça

[1] Tuba Bahtiyarca, Z., Okan, S., Şahin, S. B., & Ocak, F. (2021). Stres Kırıklarına Yaklaşım: Üç Olgu Sunumu ve Literatürün Gözden Geçirilmesi. *Turkish Journal of Osteoporosis/Turk Osteoporoz Dergisi*, *27*(2).DOI: 10.4274/tod.galenos.2020.86648

[2] Olczak, J., Fahlberg, N., Maki, A., Razavian, A. S., Jilert, A., Stark, A., … Gordon, M. (2017). Artificial intelligence for analyzing orthopedic trauma radiographs. *Acta Orthopaedica*, *88*(6), 581–586. https://doi.org/10.1080/17453674.2017.1344459

[3] Raghavendra, U.; Bhat, N.S.; Gudigar, A.; Acharya, U.R. Automated system for the detection of thoracolumbar fractures using a CNN architecture. Future Gener. Comput. Syst. 2018, 85, 184–189. http://doi.org/10.1016/j.future.2018.03.023

[4] Tobler, P.; Cyriac, J.; Kovacs, B.K.; Hofmann, V.; Sexauer, R.; Paciolla, F.; Stieltjes, B.; Amsler, F.; Hirschmann, A. AI-based detection and classification of distal radius fractures using low-effort data labeling: Evaluation of applicability and effect of training set size. Eur. Radiol. 2021, 31, 6816–6824. http://doi.org/10.1007/s00330-021-07811-2

[5] Uysal, Fatih, Fırat Hardalaç, Ozan Peker, Tolga Tolunay, and Nil Tokgöz. 2021. "Classification of Shoulder X-ray Images with Deep Learning Ensemble Models" *Applied Sciences* 11, no. 6: 2723. https://doi.org/10.3390/app11062723

[6] Thian, Y.L.; Li, Y.; Jagmohan, P.; Sia, D.; Chan, V.E.Y.; Tan, R.T. Convolutional Neural Networks for Automated Fracture Detection and Localization on Wrist Radiographs. Radiol. Artif. Intell. 2019, 1, e180001. http://doi.org/10.1148/ryai.2019180001

[7] Jain, B., Malik, D., Jagota, G. *et al.* iA-HLD: an improved AlexNet for hairline fracture detection in orthopedic images. *Neural Comput & Applic* (2024). https://doi.org/10.1007/s00521-024-10348-2

[8] Hardalaç, F.; Uysal, F.; Peker, O.; Çiçeklidağ, M.; Tolunay, T.; Tokgöz, N.; Kutbay, U.; Demirciler, B.; Mert, F. Fracture Detection in Wrist X-ray Images Using Deep Learning-Based Object Detection Models. *Sensors* 2022, *22*, 1285. https://doi.org/10.3390/s22031285

[9] Wang, W., Huang, W., Lu, Q. *et al.* Attention mechanism-based deep learning method for hairline fracture detection in hand X-rays. *Neural Comput & Applic* 34, 18773–18785 (2022). https://doi.org/10.1007/s00521-022-07412-0

[10] Wang, R., Zhao, J., Liu, X. *et al.* AdvYOLO: Advanced YOLOv8 Application for Bone Pathology Localization and Classification in Wrist X-ray Images, 29 March 2024, PREPRINT (Version 1) available at Research Square https://doi.org/10.21203/rs.3.rs-4051336/v1

Özdeğerlendirme Formu

1. **Projeniz tasarım boyutu nedir (prototip gerçekleştirme, benzetim veya analiz)?**
   Projemiz yazılım tabanlıdır ve derin öğrenme yöntemleri ile X-Ray kemik görüntüleri üzerinden kırık tespitini amaçlamaktadır. Donanım kullanımı içermemektedir.

   Proje; veri seti oluşturma, yazılım geliştirme ve bilgisayar ortamında kullanılabilecek bir kullanıcı arayüzü ile derin öğrenme modelinin entegrasyonunu içermektedir.

   Veri seti oluşturma işlemi, görüntülerin elde edilmesinden sonra her birisi için ayrı ayrı etiketleme işlemi yapmayı içerir.

   Derin öğrenme ve model oluşturma işlemleri Google Colab ortamında, kullanıcı arayüzü oluşturma işlemleri ise Virtual Studio Code ortamında ve bu iki işlem için Python programlama dili kullanılarak gerçekleştirilmiştir.

2. **Kullandığınız tasarım yöntemini/yöntemlerini açıklayınız.**
   Proje yazılım tabanlı olduğu için kullanılan tasarım yöntemleri daha çok yazılım ve sağlık standartları ve tasarım yöntemleri özelinde olmuştur.

   Veri setindeki görüntülerin etiketlenmesi için “makesense.ai” aracı kullanılmıştır.

   İlgili görüntülerde kırık olup olmadığını ve konumunu tespit edecek olan derin ağ modelleri için 4 farklı ağ modeli denenmiştir. Bunlar sırasıyla YOLO, RT-DETR, Faster R-CNN ve RetinaNet’tir. İlgili derin ağ modellerinin eğitimi için, Google tarafından sağlanan ücretsiz bulut tabanlı yazılım platformu Google Colab kullanılmıştır. Colab ortamındaki ücretsiz GPU’ların yetersizliğinden dolayı Google Colab Pro+ üyeliği satın alınmış ve daha yüksek kapasiteli GPU’lar ile birlikte eğitimler ve performans testleri gerçekleştirilmiştir.

   Eğitilen derin ağ modellerinin kullanıcı tarafından rahat biçimde kullanılabilmesi için oluşturulan kullanıcı arayüzü, Flask kütüphanesi ve Python programlama dili kullanılarak HTML tabanlı olacak şekilde tasarlanmıştır.

   Diyagram, grafik, tablo gibi şemaları oluşturmak için Canva, Excel, Microsoft Office gibi açık kaynaklı araçlar kullanılmıştır. Sunum hazırlanması için Canva, Microsoft PowerPoint araçları kullanılmıştır. Raporlar, Microsoft Word aracında yazılmıştır.

3. **Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir?**
   ISO/IEC 20546:2019 – Big Data
   ISO/IEC 23053:2022 – Framework for AI Systems
   IEEE 2801-2022 - Quality Management of Datasets for Medical Artificial Intelligence
   ISO 14971 – Tıbbi Cihazlar için Risk Yönetimi

4. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir?

Maliyet Analizi: Projenin tasarım ve üretim süreçlerinde, derin ağın eğitimi için görüntü temininde kullanılan "FracAtlas", "MURA" gibi veri setleri ve "Kaggle" internet sitesi açık kaynaklı olduğundan veri edinme konusunda bir maliyet bulunmamaktadır. Derin öğrenme için denemeler yapılırken Google Colab Pro+ sürümü kullanılacağından dolayı 4960,8TL (820,8x6) maliyet bulunmaktadır. Projenin geliştirilmesi ve devamlılığı için 480000TL iş gücü maliyeti ve 20000 TL donanım maliyeti bulunmaktadır.

Ekonomi: Derin ağların eğitim aşamasında yüksek performansa sahip ekran kartlı bilgisayarlara ihtiyaç duyulur. Bu kapsamda derin ağın eğitim işlemi için Google Colab Pro+ kullanılacaktır. Ayrıca derin ağın eğitimi sırasında kullanılacak olan X-Ray görüntülerinin yüksek kalitede olması gerekmektedir. Kaliteli röntgen cihazlarının yüksek maliyetli olması bu teknolojinin yaygın kullanımını olumsuz yönde etkileyebilir. Bu görüntülerin eğitim için etiketlenmesi de ayrıca bir maliyet oluşturmaktadır.

Sağlık: Kırıkların tespiti ile ilgili olarak alınacak kararlar hasta bireylerin tedavi süreçleri üzerinde doğrudan etkili olacaktır. Bu nedenle tıbbi teşhislerde hata payı oldukça düşük olmalıdır. Derin ağın eğitimi sırasında yüksek başarı oranına ulaşılıp, oluşturulan sistemin yüksek hassasiyet ve güvenilirliğe sahip olması hedeflenmelidir. Çıkarılan yanlış sonuçlar hasta bireyin üzerinde yanlış tedavi veya hiç tedavi uygulanmaması gibi sonuçlara yol açabilir ve bunun da hukuki sonuçları olabilir.

5. Proje yönetimi nasıl gerçekleştirdiğinizi açıklayınız.

Proje yönetimi, iki kişilik bir ekip tarafından planlı ve aşamalı bir şekilde yürütülmüştür. Çalışmalar, belirlenen proje takvimine uygun olarak ilerletilmiş ve düzenli toplantılarla ilerleme değerlendirilmiştir.

- Veri Toplama ve Etiketleme (Süre: 6 Hafta, İş Yükü: %40)

İbrahim Halil Atay (%60) Emir Kıvrak (%40)

- Model Geliştirme ve Eğitim (Süre: 8 Hafta, İş Yükü: %35)

İbrahim Halil Atay (%60) Emir Kıvrak (%40)

- Test, Analiz ve Sonuçların Değerlendirilmesi (Süre: 2 Hafta, İş Yükü: %15)

İbrahim Halil Atay (%30) Emir Kıvrak (%70)

- Rapor Yazımı (Süre: 4 Hafta, İş Yükü: %10)

İbrahim Halil Atay (%20) Emir Kıvrak (%80)

6. Öneri raporunda yapılan takvime ve/veya konuya gore değişiklik/sapma oldu mu?

Evet oldu. "Stres Kırığı Tespiti" olarak belirlenen başlık, "Kırık Tespiti" olarak düzenlendi. Bunun nedeni, açık kaynak veri setlerinde, model eğitiminde kullanılmak üzere stres kırığı görüntülerinin yeteri sayıda olmaması ve model eğitiminin düzgün yapılamamasıdır.

İncelenen çalışmalarda, kemik X-Ray görüntüleri üzerinde kırık tespiti için yapılan nesne algılama (object detection) çalışmalarında, önem verilen başarı metriği olan mAP50 değerinin %70 civarlarında olması sebebiyle "Hedefler" kısmındaki başarı hedefi +%70 mAP50 olarak güncellenmiştir.

7. Proje çalışması sırasında ne tür problemler ve sorunlarla karşılaştınız?

Gerçekleştirilen model eğitimlerinde, kırık olmayan X-Ray kemik görüntüleri modele öğretilmediği zaman, eklem boşluklarının kırık olarak tespit edilmesi şeklinde bir sorun yaşandı. Bu probleme çözüm olarak veri setine kırık olmayan X-Ray kemik görüntüleri de eklendi.

Eğitilen modelde kullanılan veri setinde, her bölgeden eşit sayıda veri bulunmamaktadır. Bu sebeple model, bilek ve kol fotoğraflarındaki kırıkları çok yüksek bir başarı oranı ile tespit edebilirken; omuz, ayak ve bacak gibi bölgelerdeki X-Ray görüntülerinde başarısız sonuçlar çıkarmaktadır. Bu da sonuç olarak modelin başarı oranını yüksek oranda azaltmaktadır. Bunu engellemek amacıyla veri sayısı arttırılıp vücut bölgeleri arasında denge sağlanması amacıyla kırık görüntüleri bölgesel olarak farklı şekillerde veri artırma işlemlerine tabi tutuldu.