Bone Fracture Detection Using YOLOv8

Bone fracture detection from X-ray images is a challenging task due to low contrast, subtle fracture patterns, and high variability across patients. This project presents an automated computer vision system for detecting and localizing bone fractures in X-ray images using the YOLOv8 object detection framework.

A publicly available Kaggle dataset is used to train and evaluate the model. The trained model is deployed through a web-based interface using Gradio, enabling interactive inference. The system is designed for educational and research purposes and demonstrates the practical application of deep learning in medical image analysis.

1. Introduction

Medical imaging plays a critical role in fracture diagnosis, yet manual interpretation of X-ray images is time-consuming and prone to inter-observer variability. Recent advances in deep learning, particularly convolutional neural networks (CNNs), have enabled automated image analysis with promising results.

Object detection models such as YOLO (You Only Look Once) offer real-time detection capabilities, making them suitable for rapid screening systems. This project aims to design, train, and deploy a YOLOv8-based model to detect bone fractures in X-ray images. The focus is on demonstrating an end-to-end pipeline, including dataset preparation, model training, evaluation, and deployment.

2. Dataset Description

The dataset used in this project is the *Bone Fracture Detection – Computer Vision Project* dataset obtained from Kaggle. It consists of X-ray images of bones with annotated fracture regions.

Source

Kaggle

Data Type

X-ray images

Annotations

Bounding boxes indicating fracture locations

Task Type

Object detection

The dataset is organized in YOLO format, with separate directories for training and validation images and corresponding label files. Each label file contains normalized bounding box coordinates.

3. Methodology

01

Model Selection

YOLOv8, developed by Ultralytics, was selected due to its balance between detection accuracy and computational efficiency. The nano variant (YOLOv8n) was initially used to accommodate limited computational resources. YOLOv8 performs object detection in a single forward pass, making it suitable for real-time inference. Pretrained weights on the COCO dataset were used as a starting point to improve convergence.

02

Data Configuration

A YAML configuration file was created to specify the paths to the training and validation datasets, the number of classes, and the class name (fracture). This configuration enables YOLOv8 to correctly load and interpret the dataset during training.

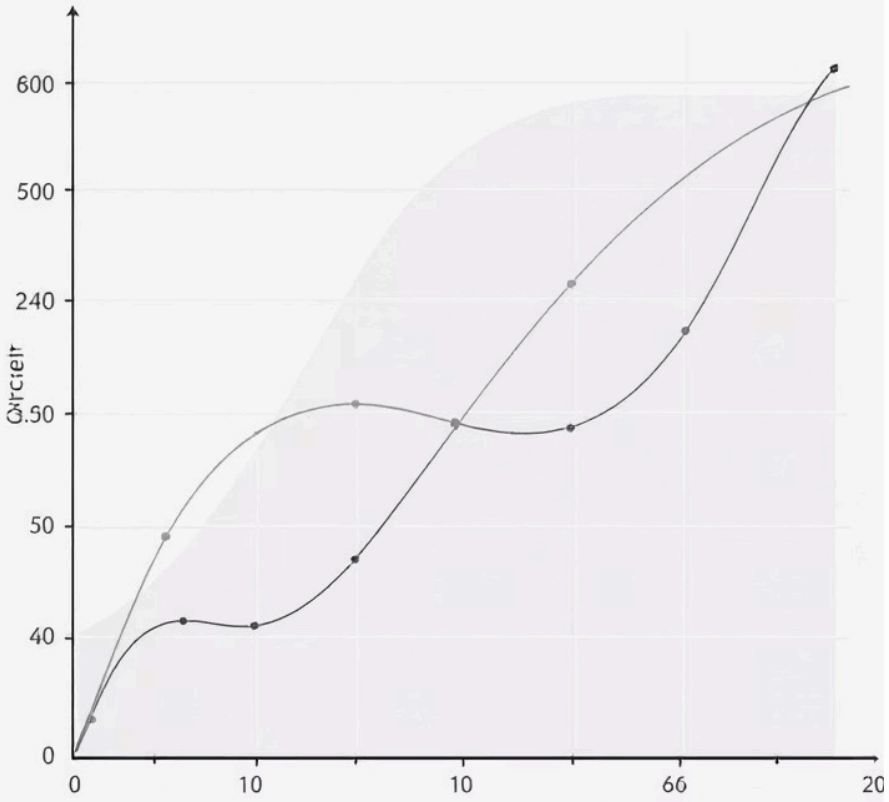
03

Training Procedure

The model was trained using the Ultralytics YOLOv8 training API with the following key parameters:

- **Epochs:** 50
- **Image size:** 640 × 640
- **Batch size:** 32
- **Optimizer:** Adam
- **Early stopping:** Enabled using patience parameter

Training outputs, including model weights and performance metrics, were automatically saved for later evaluation.



4. Model Evaluation

Model performance was evaluated on the validation dataset using standard object detection metrics:

Precision

Measures the accuracy of positive predictions.

Recall

Measures the model's ability to find all relevant instances.

Mean Average Precision (mAP)

A comprehensive metric that averages precision values across different recall thresholds.

Visual inspection of predictions was also performed by overlaying bounding boxes on validation images. While the model successfully detected prominent fractures, subtle fracture patterns remained challenging, which is consistent with known limitations of object detection models in medical imaging.

5. Deployment

To demonstrate practical usability, the trained YOLOv8 model was deployed using Gradio, a lightweight framework for building machine learning web interfaces.

1

Load Weights

The trained model weights were loaded into memory.

2

Upload Image

Users upload an X-ray image through the Gradio interface.

3

Process & Infer

The image is processed and passed to the YOLOv8 model for inference.

4

Visualize Results

Detected fracture regions are visualized using bounding boxes.

5

Display Output

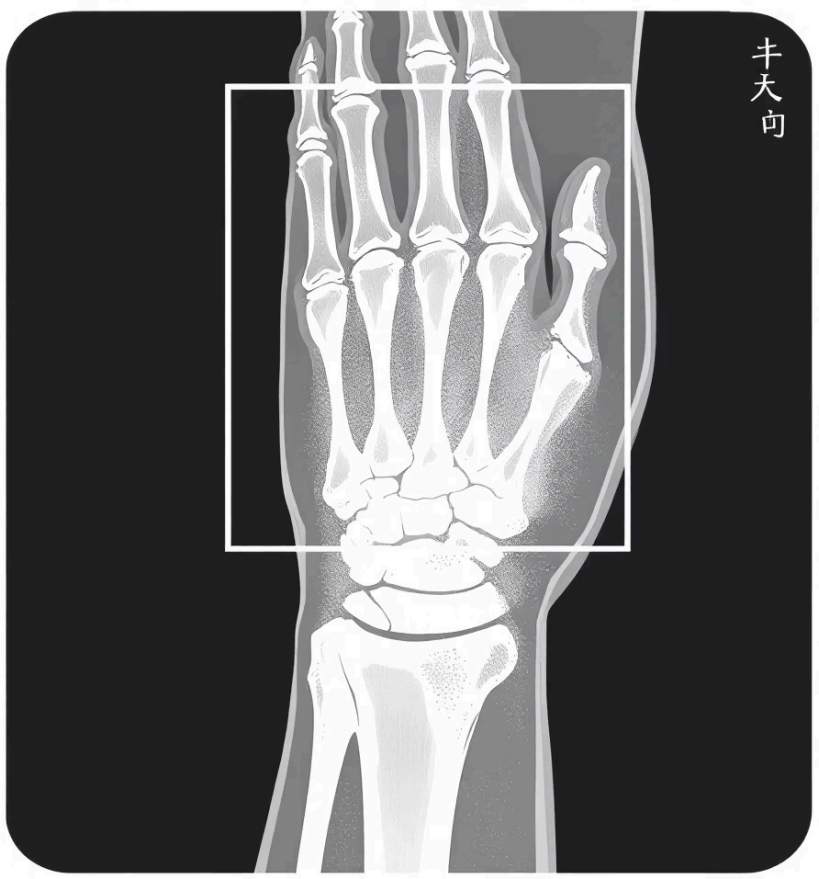
The annotated image is displayed to the user.

The deployment supports local execution and cloud-based environments such as Google Colab or Hugging Face Spaces.

6. Results and Discussion

The deployed system demonstrates the feasibility of automated fracture detection using YOLOv8. The model performs reasonably well on clear fracture cases but struggles with hairline fractures and low-contrast regions. These limitations are primarily attributed to dataset size, annotation quality, and the inherent difficulty of medical image interpretation.

Despite these challenges, the system effectively showcases an end-to-end deep learning pipeline from data acquisition to deployment.



7. Limitations

Limited Dataset

Small dataset size and potential inaccuracies in annotations.

Subtle Fractures

Difficulty in accurately detecting hairline or low-contrast fracture patterns.

No Clinical Validation

The model has not undergone formal clinical trials or validation.

Hardware Constraints

Use of a lightweight model variant (YOLOv8n) due to limited computational resources.

8. Ethical Considerations

> This system is intended strictly for educational and research purposes. It is not designed or approved for clinical diagnosis. All predictions should be interpreted cautiously and must not replace professional medical judgment.

Educational Tool

Primarily for learning and research.

No Clinical Use

Not approved for patient diagnosis.

Interpret Cautiously

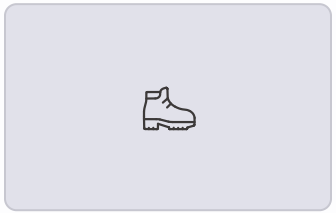
Model predictions require human oversight.

Medical Judgment

Does not replace professional expertise.

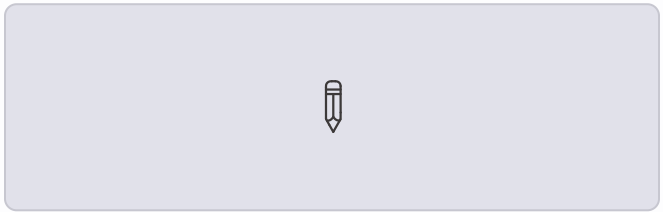
9. Conclusion

This project presents a complete workflow for bone fracture detection using YOLOv8, including training, evaluation, and deployment. The results highlight both the potential and challenges of applying object detection models to medical imaging tasks.



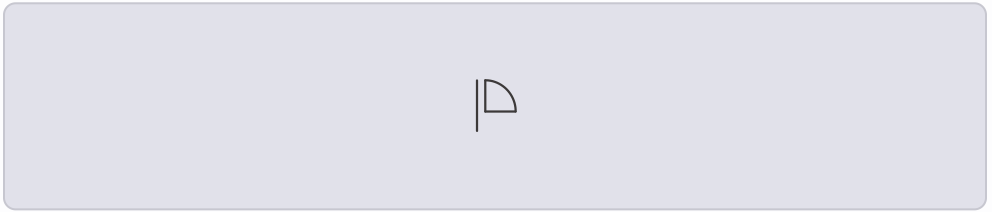
Higher Resolution Training

To capture finer details.



Improved Annotations

More precise bounding boxes.



Segmentation-Based Approaches

For pixel-level accuracy.

Future work may focus on higher-resolution training, improved annotations, and segmentation-based approaches to overcome current limitations.