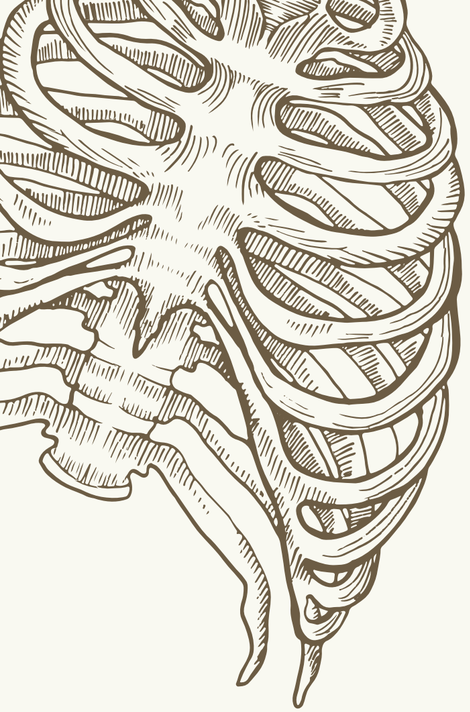
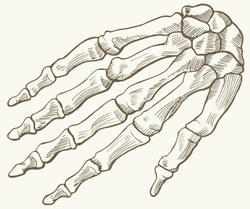
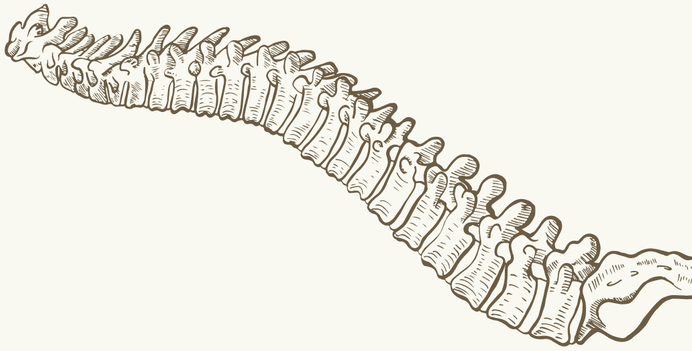


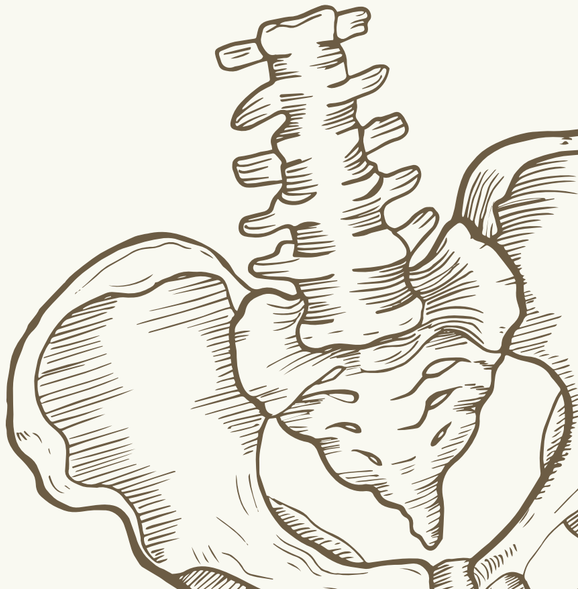


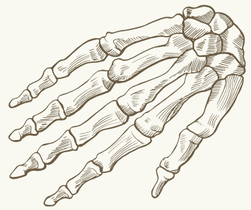


Bone Fracture Detection

Deep Learning

By: Maryam ALAamri

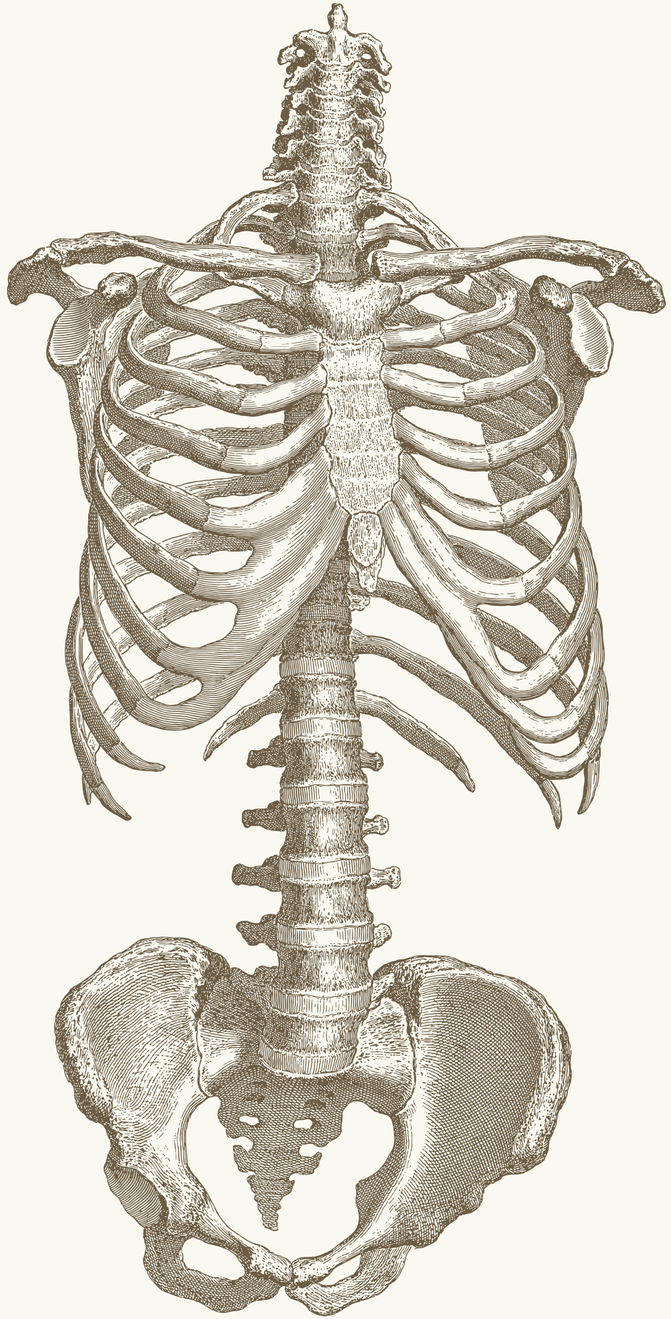


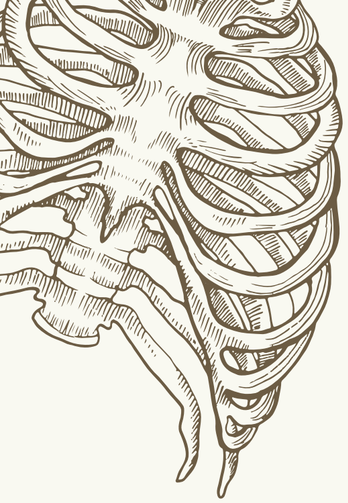


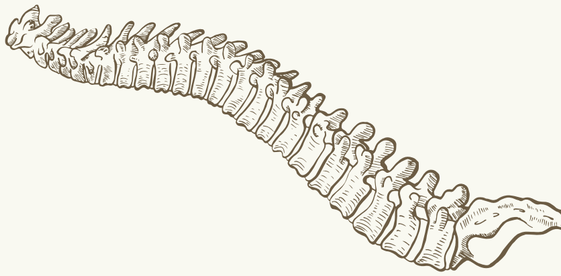
Introduction

Bone fractures are a common medical condition that require fast and accurate diagnosis.
Traditional diagnosis using X-ray images depends heavily on radiologists' experience and can be time-consuming, especially in emergency situations.
With recent advances in deep learning and computer vision, object detection models such as YOLO (You Only Look Once) enable real-time and accurate fracture localization in medical images.
YOLO models are particularly suitable because they:

- Detect objects in a single forward pass
- Provide bounding boxes and confidence scores
- Work well for real-time medical applications

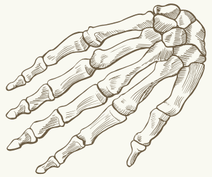






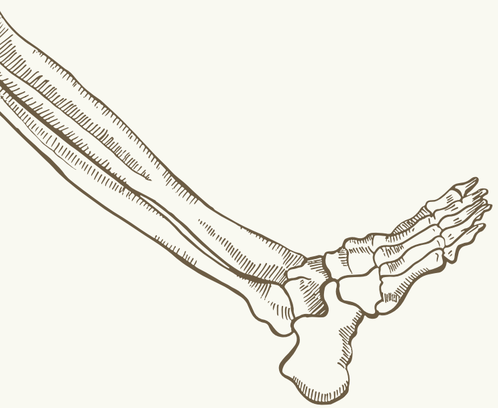
- **Defined Problem**

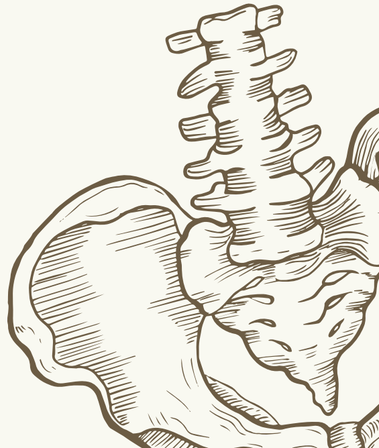
  Bone fractures are common injuries, and interpreting X-ray images requires expert radiologists. Manual diagnosis can be time-consuming, subjective, and error-prone, especially in emergency cases.

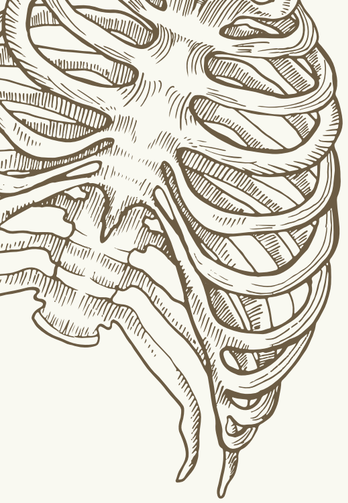


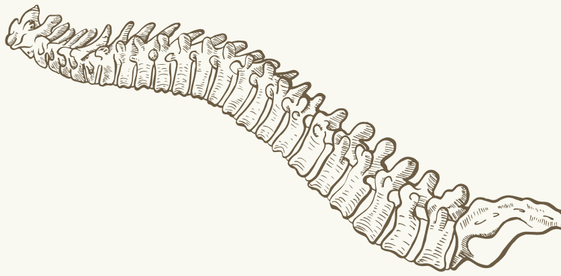
- **Problem statement**

  Automatically detect bone fractures in X-ray images using deep learning to assist doctors and improve diagnostic speed and accuracy.





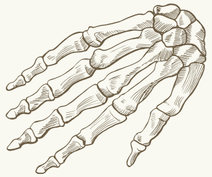


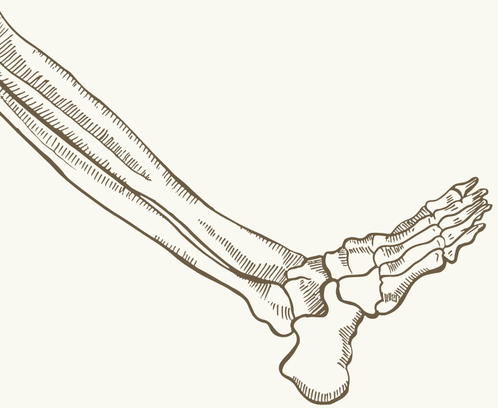


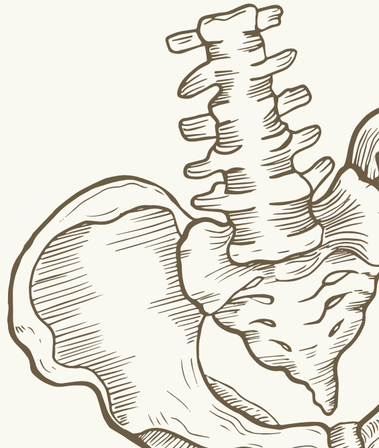
- **Objective**

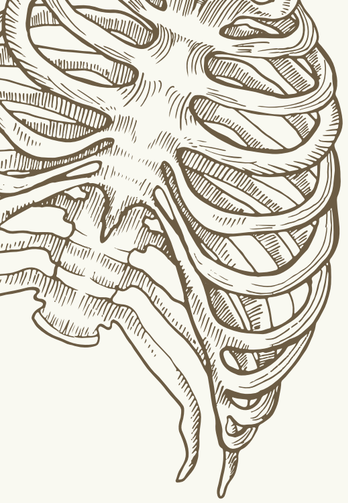
The goal of this project is to:

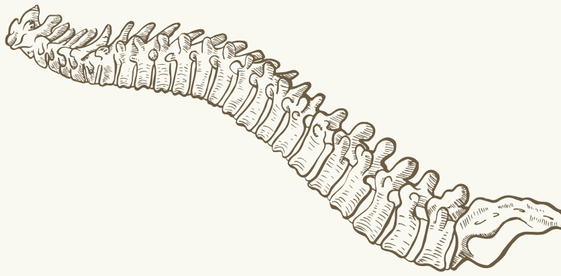
- Prepare a bone fracture detection dataset
- Train a YOLOv8 object detection model
- Evaluate performance using Precision, Recall, F1-Score, and mAP
- Visualize fracture detection using bounding boxes
- Build a real-time fracture detection system





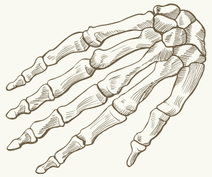
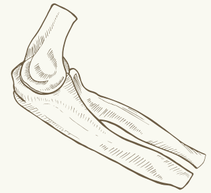
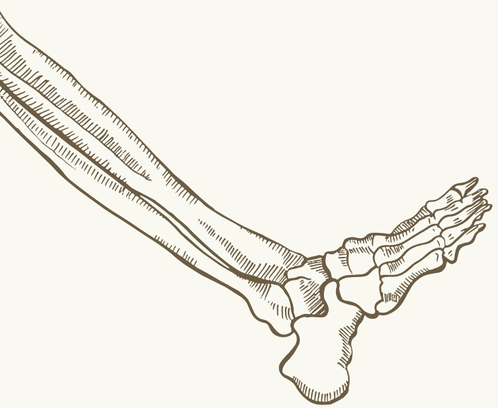


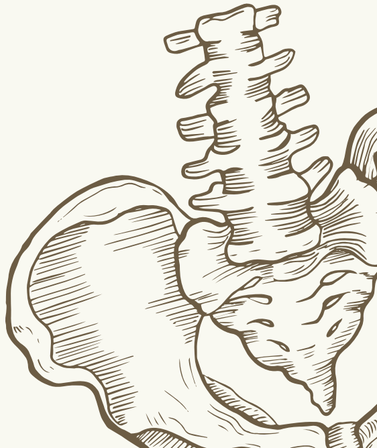




Dataset Information

Dataset Source
Kaggle
Dataset name:
Human Bone Fractures Multi-modal Image Dataset (HBFMID)
Dataset Type
X-ray images of human bones
Number of Images
1539 X-ray images
Split into:
Training set 1347
Validation set 128
Image Size
Images are resized to 640 × 640 pixels during training
Class Label
fracture (The model detects the presence and location of fractures using bounding boxes.)

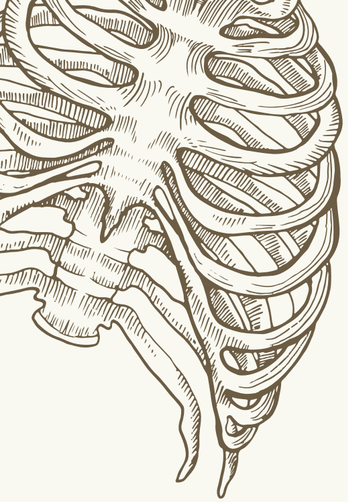




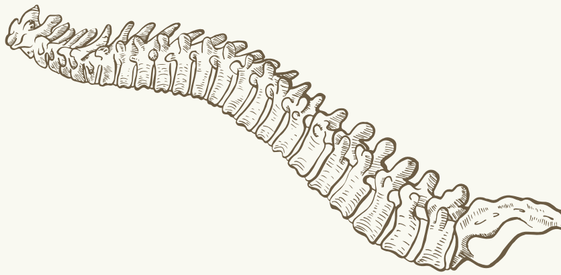


Create YOLO Database YAMAL

```python
#STEP 3: Create YOLO Dataset YAML
import yaml

data_yaml = {
    "path": DATASET_PATH,
    "train": "train/images",
    "val": "valid/images",
    "test": "test/images",
    "names": {
        0: "fracture"
    }
}

with open("bone_fracture.yaml", "w") as f:
    yaml.dump(data_yaml, f)

print("bone_fracture.yaml created")
```

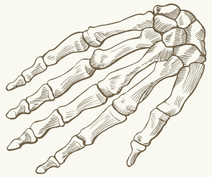
```
bone_fracture.yaml created
```

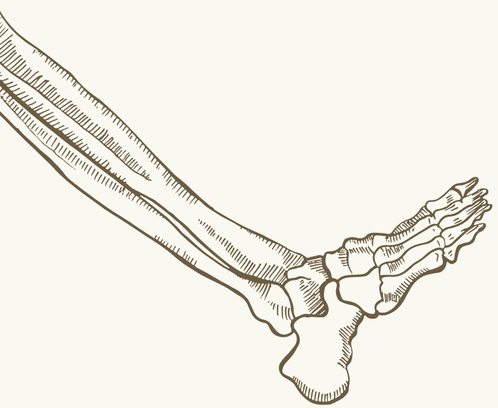


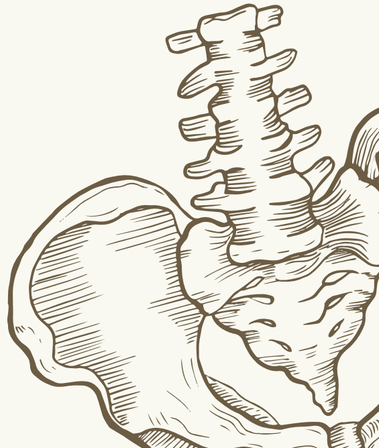


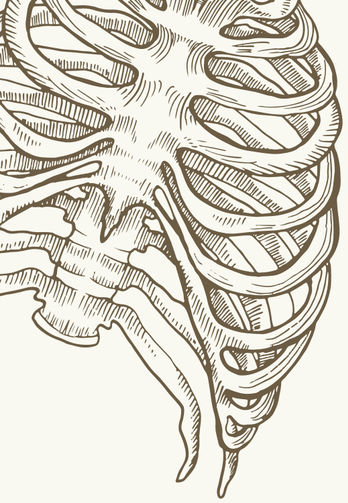
Train YOLO Model

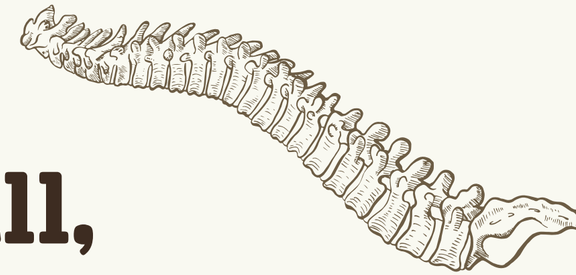
```python
#STEP 4: Train YOLO Model
from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(
    data="bone_fracture.yaml",
    epochs=50,
    imgsz=640,
    batch=16,
    device="cpu"
)
```





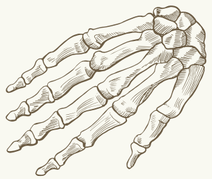


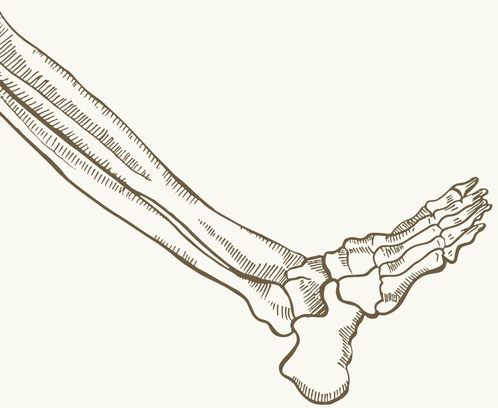


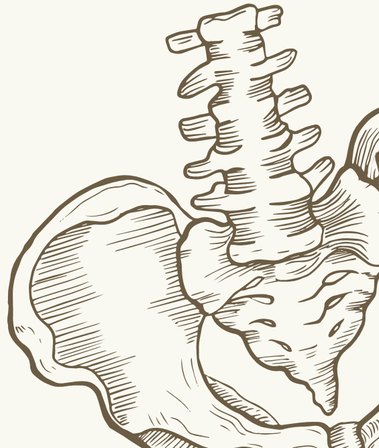


Evaluate Model (Precision, Recall, F1, mAP)

```python
from ultralytics import YOLO

# Load trained model
model = YOLO("runs/detect/train/weights/best.pt")

# Run validation
metrics = model.val()

# Correct metric extraction (YOLOv8.3+)
precision = metrics.box.mp
recall = metrics.box.mr
map50 = metrics.box.map50
map5095 = metrics.box.map
f1_score = metrics.box.f1.mean()  # average F1 across classes

# Print results
print(f"Precision : {precision:.4f}")
print(f"Recall    : {recall:.4f}")
print(f"F1 Score  : {f1_score:.4f}")
print(f"mAP@50    : {map50:.4f}")
print(f"mAP@50:95 : {map5095:.4f}")
```



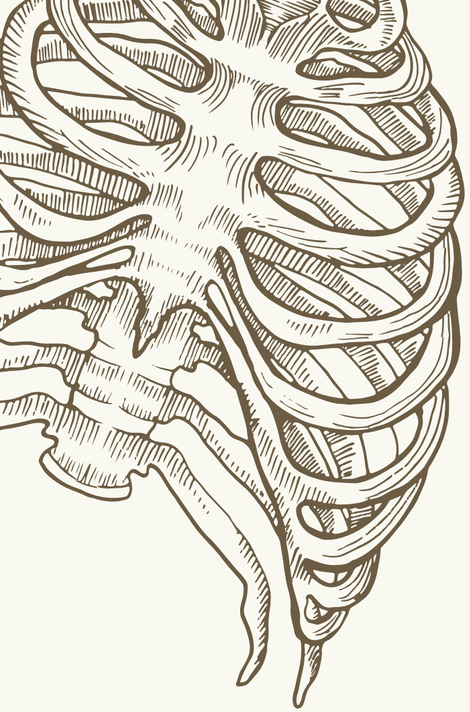


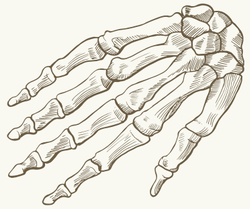


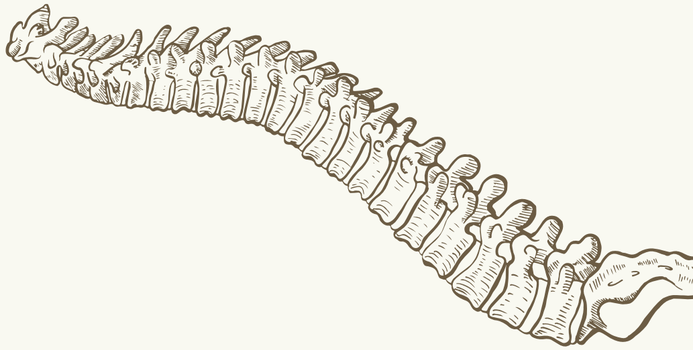
Interpretation, evaluation of the model

- Precision = 99.6%: The model produced almost no false-positive fracture detections.
- Recall = 71.4%: Most fractures were correctly detected, though a number of subtle cases were still missed.
- F1-score = 83.2%: This shows a strong overall balance between precision and recall.
- mAP@50 = 76.9%: The model demonstrates good object-localization performance at the standard IoU threshold.
- Lower mAP@50–95: Performance drops under stricter IoU thresholds, reflecting the challenge of precisely localizing thin or faint fracture lines.

```
Precision : 0.9957
Recall    : 0.7143
F1 Score  : 0.8318
mAP@50    : 0.7695
mAP@50:95 : 0.4753
```

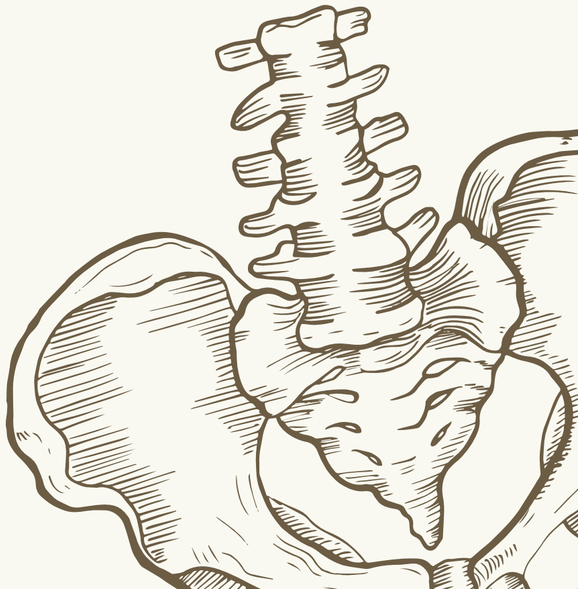






Visualize Predictions

```
#STEP 6: Visualize Predictions (Bounding Boxes)
import matplotlib.pyplot as plt
import cv2
import glob

test_images = glob.glob(f"{DATASET_PATH}/test/images/*.jpg")

results = model.predict(test_images[:2], conf=0.4)

for r in results:
    img = r.plot()
    plt.figure(figsize=(6,6))
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.axis("off")
```



Visualize Predictions

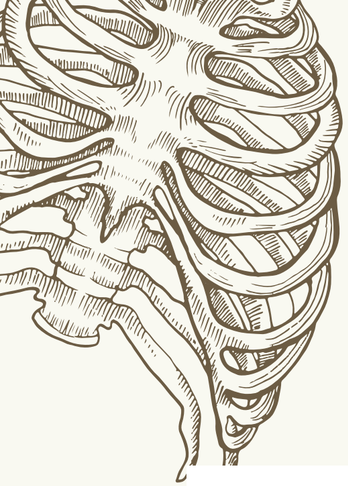
```
0: 640x640 (no detections), 198.9ms
1: 640x640 (no detections), 198.9ms
```

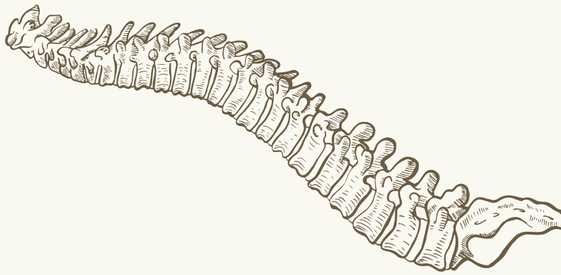
Interface using Gradio

```python
import gradio as gr
import cv2
import numpy as np
from ultralytics import YOLO

# 🔹 Load YOUR trained model (IMPORTANT)
model = YOLO("runs/detect/train/weights/best.pt")

def fracture_detector(image):
    """
    image: PIL image from Gradio
    """

    # Convert PIL image to OpenCV format
    img = np.array(image)
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

    # Run YOLO inference
    results = model.predict(
        img,
        conf=0.3,
        iou=0.5,
        verbose=False
    )

    # Get detected boxes
    boxes = results[0].boxes
    fractured = boxes is not None and len(boxes) > 0

    # Draw bounding boxes
    annotated_img = results[0].plot()
    annotated_img = cv2.cvtColor(annotated_img, cv2.COLOR_BGR2RGB)

    # Diagnosis output
    diagnosis = "🟥 Fractured Bone Detected" if fractured else "🟩 No Fracture Detected"

    return annotated_img, diagnosis
```

```python
# 🔹 Gradio Interface
demo = gr.Interface(
    fn=fracture_detector,
    inputs=gr.Image(type="pil", label="Upload X-ray Image"),
    outputs=[
        gr.Image(label="Detection Result"),
        gr.Textbox(label="Diagnosis")
    ],
    title="Bone Fracture Detection System (YOLOv8)",
    description=(
        "This system uses a YOLOv8 deep learning model trained on X-ray images "
        "to detect bone fractures in real time."
    )
)

# 🔹 Launch app
demo.launch(share=True)
```

Interface using Gradio





gging Face Spaces (https://huggingface.co/spaces)

Bone Fracture Detection System (YOLOv8)

This system uses a YOLOv8 deep learning model trained on X-ray images to detect bone fractures in real time.

Upload X-ray Image

Drop Image Here

- or -

Click to Upload

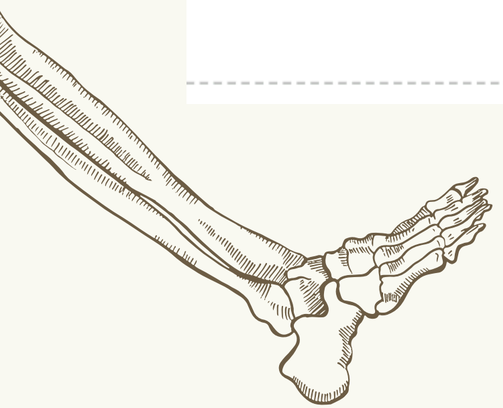
Clear

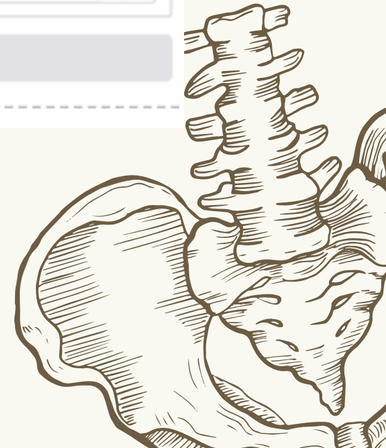
Submit

Detection Result

Diagnosis

Flag



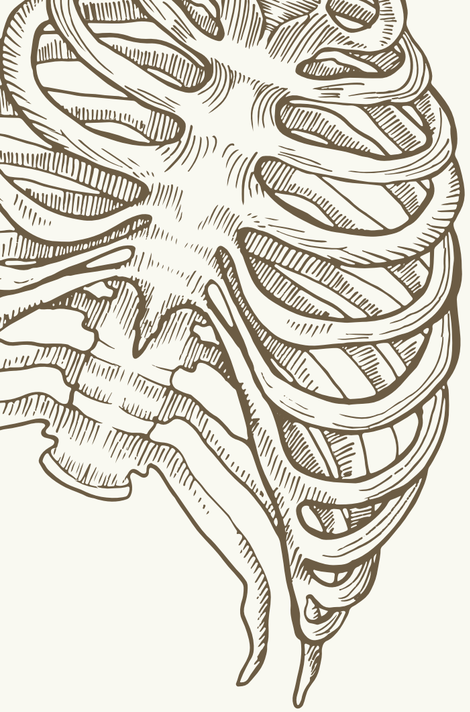


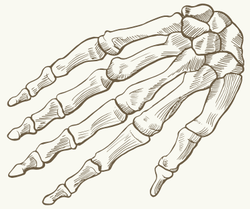
Test the model

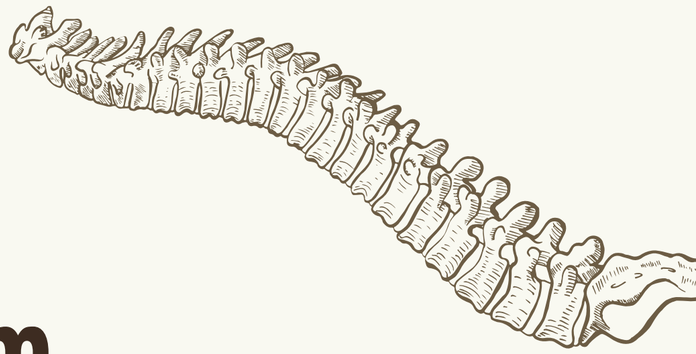
Test the model

Test the model

Test the model



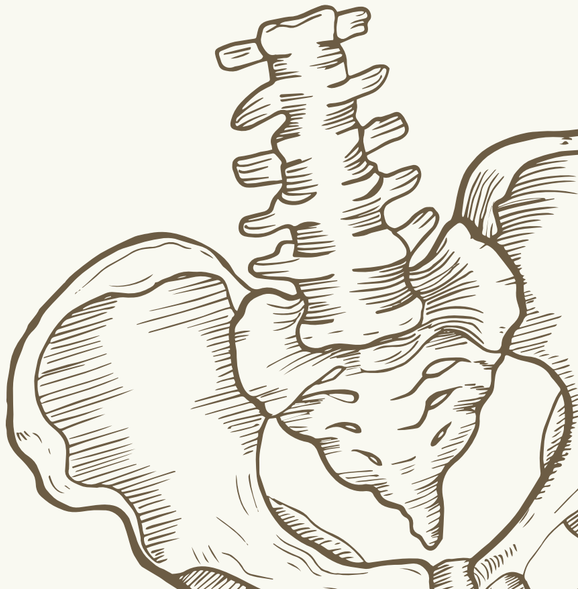


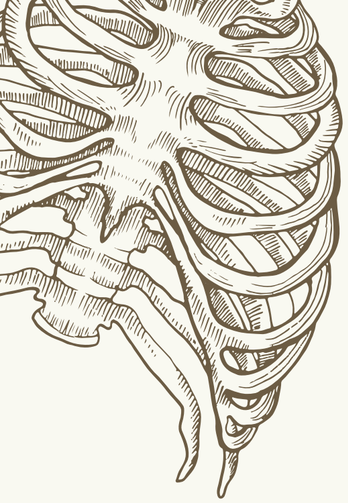


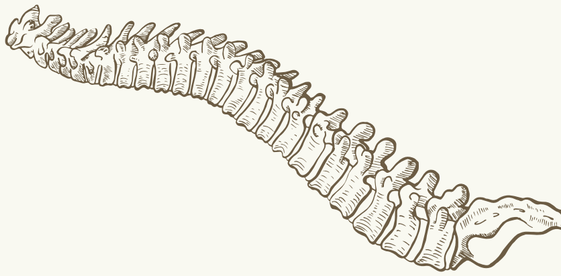
Who Will Use This System

This system can be used by:

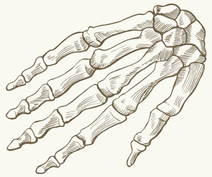
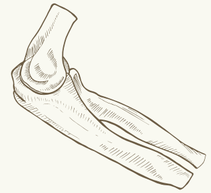
- Radiologists – as a decision-support tool
- Doctors in emergency departments – for fast fracture screening
- Hospitals & clinics – to reduce workload and diagnosis time
- Medical AI researchers & students – as a baseline fracture-detection model





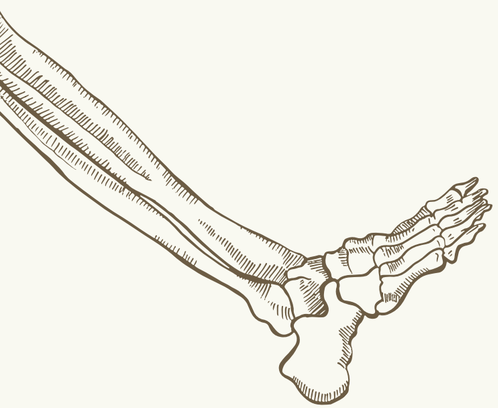


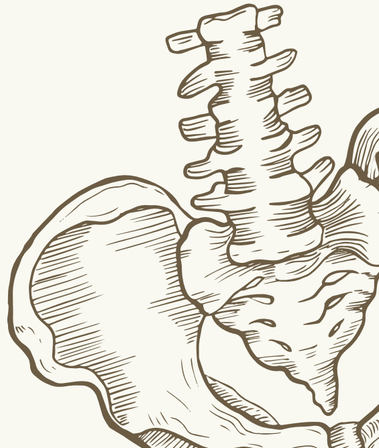
Conclusion

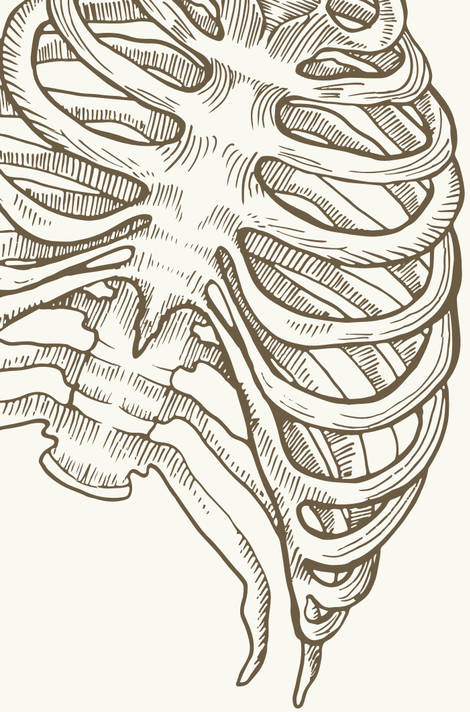



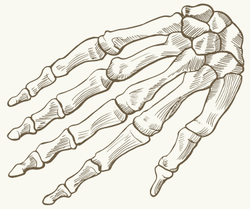
YOLOv8 demonstrates strong potential for medical image analysis, especially in time-critical scenarios such as emergency diagnosis.

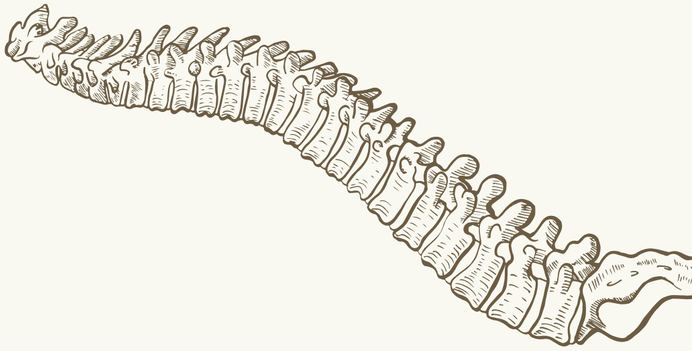
With further training, more data, and multi-class labeling, the system can be extended to detect different fracture types and support clinical decision-making.











Thank You