**Assignment 1: Real-world Image Enhancement and Analysis**

**1. Introduction (The Problem)**

Bone fractures are often diagnosed using X-ray images, but many images suffer from low contrast, poor lighting, or obscured details. These limitations can make it difficult for medical professionals to accurately identify fractures, particularly in subtle cases. The goal of this project is to enhance X-ray images of bones to improve visibility of fracture lines and internal bone structures, aiding in more reliable diagnosis.

**2. Dataset**

* **Source:** Publicly available bone fracture X-ray dataset from Kaggle   
  <https://www.kaggle.com/datasets/pkdarabi/bone-fracture-detection-computer-vision-project>
* **Content:** X-ray images of various bones with annotated fractures.
* **Format:** JPEG/PNG images, grayscale, varying resolutions (commonly 512×512 or 1024×1024).
* **Problem for Enhancement:** Many images are low-contrast, especially in regions around subtle fractures. Shadows and bright regions may obscure fracture lines.

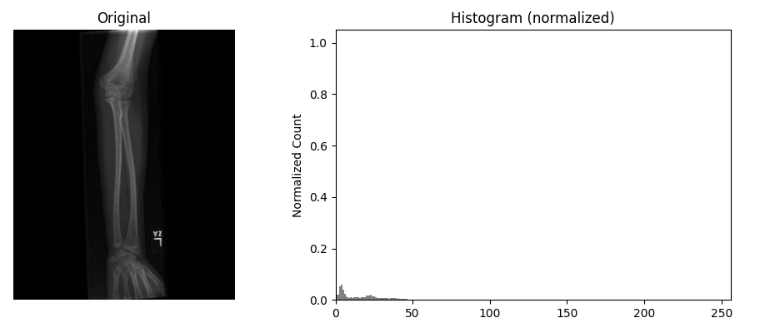
**3. Methodology & Justification**

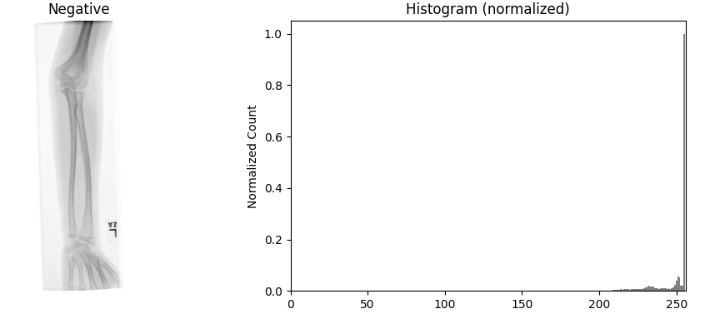
**Technique 1: Negative Image Transformation**

* **Justification:** Enhances details in dark regions by inverting intensity levels. Fracture lines often appear darker than surrounding bone; inverting can highlight these subtle details.
* **Transformation Function:**

s=L-1-r

Where L is the maximum grey level (255) and r is the original pixel intensity.



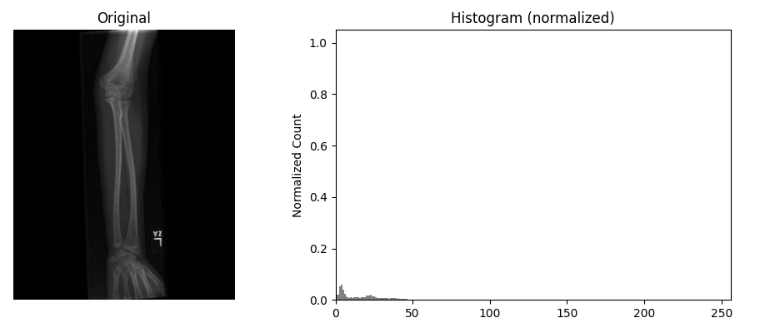


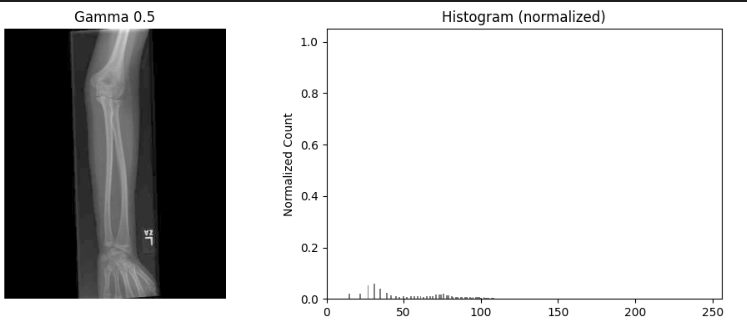
**Technique 2: Power-Law (Gamma) Transformation**

* **Justification:** Useful for adjusting overall brightness and contrast. Can brighten shadows (dark fracture areas) without overexposing bright regions.
* **Transformation Function:**

s=c.r^ γ

* + γ < 1 brightens darker regions. Selected to reveal subtle fractures without washing out bone details.





**4. Results & Analysis**

* **Visual Comparison:**

* **Histogram Analysis:**
  + Negative Transformation: Histogram is inverted; dark areas are highlighted.
  + Gamma Transformation: Histogram is stretched toward brighter intensities.
  + Combined: Enhanced visibility with broader intensity range; histogram reflects both inversion and brightness adjustment.
* **Critical Evaluation:** Discuss which technique worked best for subtle fractures and why.

**5. Conclusion**

* Both negative and gamma transformations improved visibility of bone fractures.
* Combined approach is most effective for subtle fractures in low-contrast regions.
* Recommendation: Apply gamma correction after negative transformation for enhanced diagnostic support.

**6. Appendix (Code)**

#!/usr/bin/env python3

import argparse

import os

import cv2

import numpy as np

def negative\_transform(img):

    """Apply negative transformation."""

    return 255 - img

def gamma\_transform(img, gamma=0.5):

    """Power-law (gamma) transformation."""

    img\_float = img / 255.0

    img\_gamma = np.power(img\_float, gamma)

    return np.uint8(img\_gamma \* 255)

def main():

    parser = argparse.ArgumentParser(description="Image Enhancement Script")

    parser.add\_argument("input", help="Path to input image")

    args = parser.parse\_args()

    img = cv2.imread(args.input)

    if img is None:

        raise FileNotFoundError(f"Cannot read image at {args.input}")

    print("Choose enhancement to apply:")

    print("1. Negative")

    print("2. Gamma")

    print("3. Both Negative + Gamma")

    choice = input("Enter 1, 2, or 3: ").strip()

if choice == "1":

        enhanced = negative\_transform(img)

    elif choice == "2":

        gamma\_val = float(input("Enter gamma value (e.g., 0.5, 1.5): "))

        enhanced = gamma\_transform(img, gamma=gamma\_val)

    elif choice == "3":

        gamma\_val = float(input("Enter gamma value for Gamma transform: "))

        enhanced = gamma\_transform(img, gamma=gamma\_val)

        enhanced = negative\_transform(enhanced)

    else:

        print("Invalid choice. Exiting.")

        return

    in\_dir = os.path.dirname(os.path.abspath(args.input))

    out\_dir = os.path.join(in\_dir, "output")

    os.makedirs(out\_dir, exist\_ok=True)

    filename = os.path.basename(args.input)

    name, ext = os.path.splitext(filename)

    out\_path = os.path.join(out\_dir, f"enh\_{name}{ext}")

    cv2.imwrite(out\_path, enhanced)

    print(f"Enhanced image saved to: {out\_path}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()