IMAGE PROCESSING& COMPUTER VISION: FRACTURED BONES

by Julia Nuss

PROJECT OVERVIEW

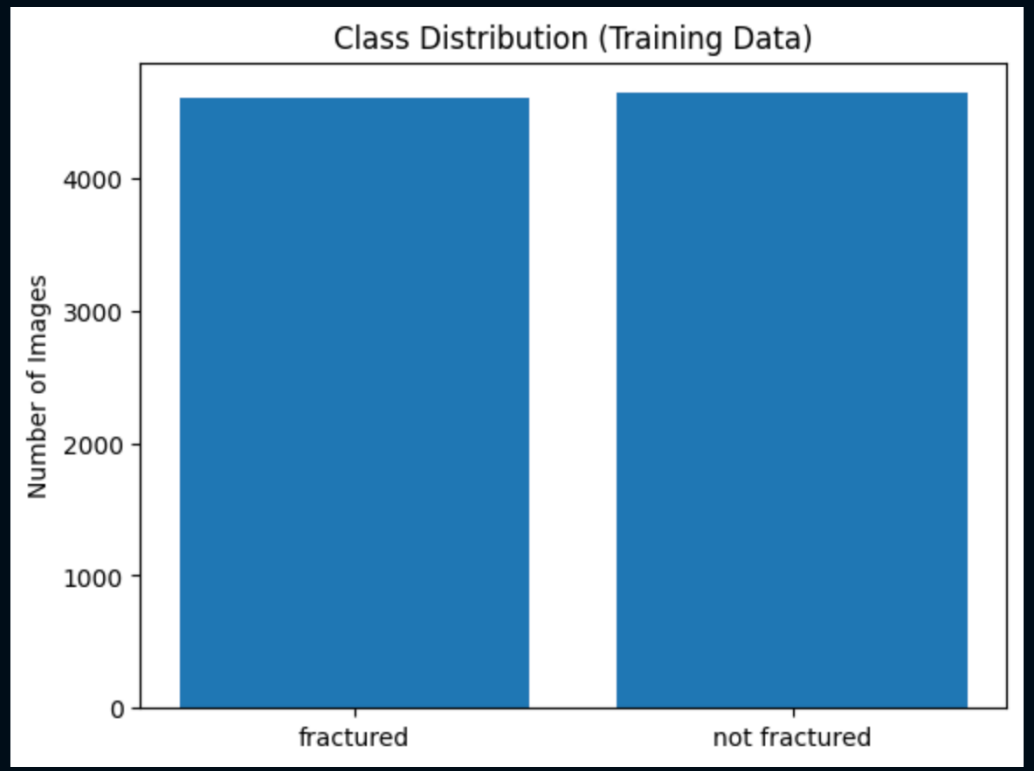
MAIN GOALS:

- Analyzing the dataset
- Creating a model that can classify bones as fractured or not fractured
- Evaluating the model
- Deploying the model

IMPLEMENTATION DETAILS

- EDA: Analyzing the dataset

```
Class Distribution in Training Set: {'fractured': 4606, 'not fractured': 4640}
```




```
Found 9246 images belonging to 2 classes.
Found 829 images belonging to 2 classes.
Found 506 images belonging to 2 classes.
```

```
"""
Bone_Fracture_Binary_Classification/
├── train/
│   ├── fractured/
│   └── not fractured/
├── test/
│   ├── fractured/
│   └── not fractured/
└── val/
    ├── fractured/
    └── not fractured/
"""
```

fractured

fractured

fractured

fractured

fractured

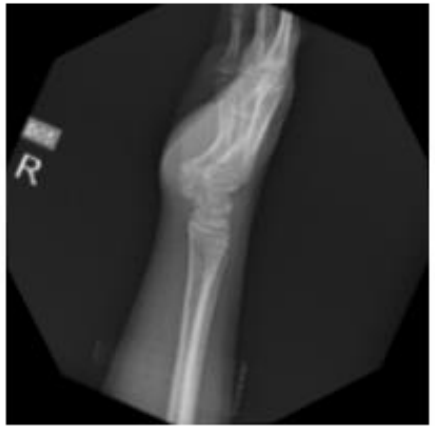
not fractured

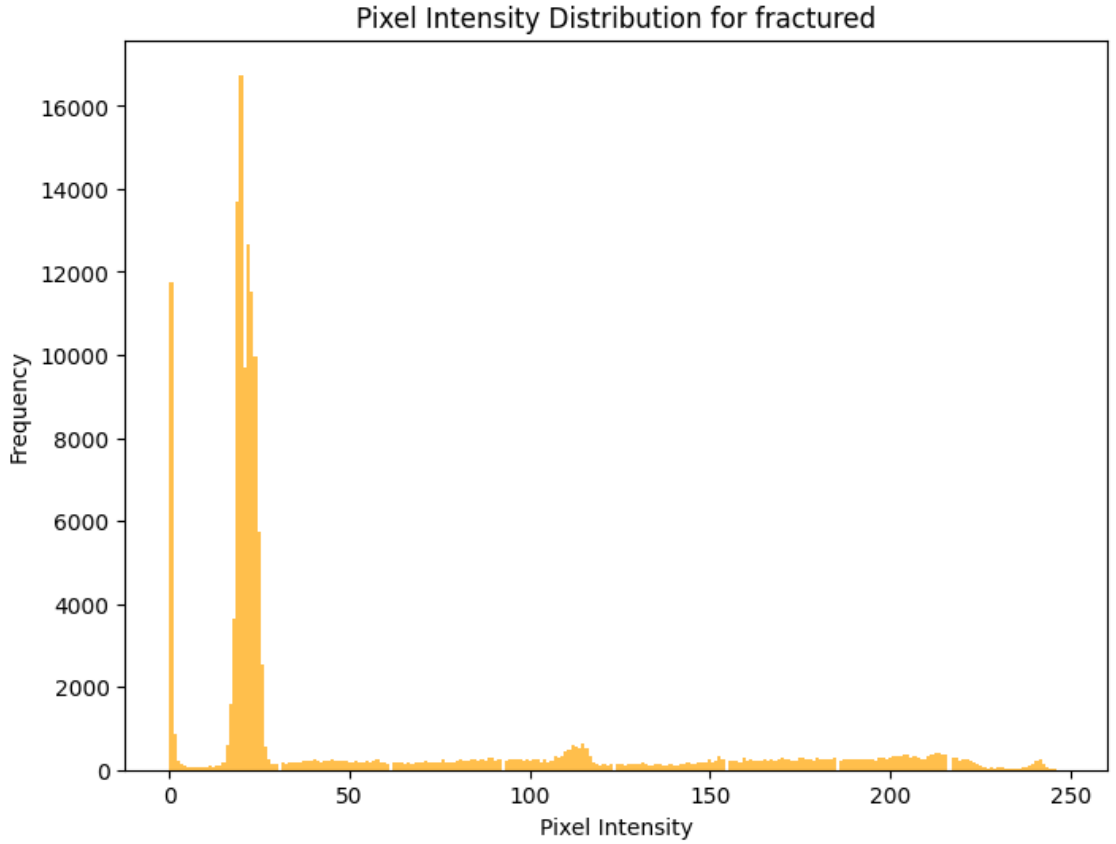
not fractured

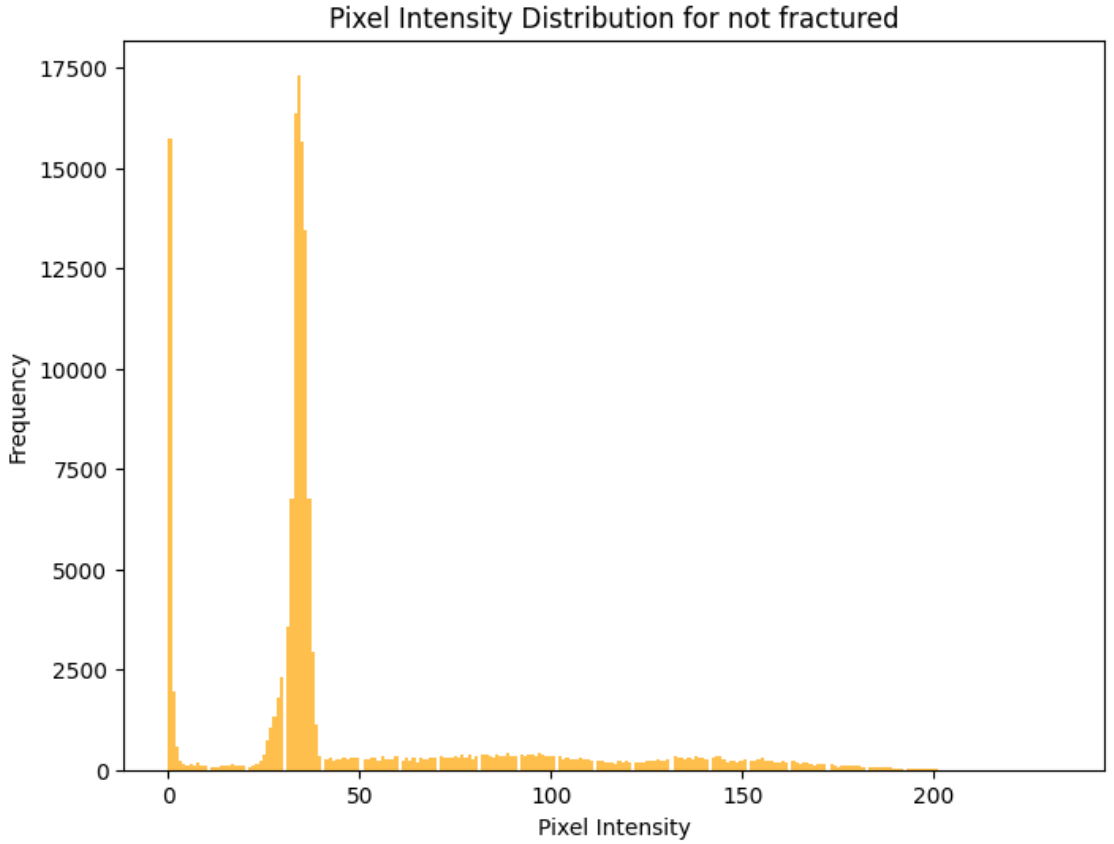
not fractured

not fractured

not fractured




Pixel Intensity Distribution for fractured
Frequency
Pixel Intensity
0
2000
4000
6000
8000
10000
12000
14000
16000
0
50
100
150
200
250


Pixel Intensity Distribution for not fractured
Frequency
Pixel Intensity
0
2500
5000
7500
10000
12500
15000
17500
0
50
100
150
200

IMPLEMENTATION DETAILS

- IMAGE PREPROCESSING:
  - Grayscaling the images to make sure they only have one channel
  - Reshaping & Rescaling
  - Horizontal Flip
  - Normalizing the images

```python
# Resize and normalize grayscale images
def preprocess_image(img, target_size=(224, 224)):
    # Ensure the image is grayscale (single channel)
    if len(img.shape) == 3:  # If the image is not grayscale
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Resize the image to the desired size
    img_resized = cv2.resize(img, target_size)

    # Normalize the pixel values to [0, 1] AFTER resizing
    img_normalized = img_resized / 255.0

    # Expand dimensions to match the expected input shape (height, width, channels)
    # Replicate the grayscale channel to simulate 3 channels (e.g., (224, 224, 3))
    img_expanded = np.stack([img_normalized]*3, axis=-1)  # Stack to create 3 channels

    return img_expanded

# Augmenting data with ImageDataGenerator
train_datagen = ImageDataGenerator(
    rotation_range=20,         # Rotate images for augmentation
    width_shift_range=0.1,     # Horizontal shift
    height_shift_range=0.1,    # Vertical shift
    zoom_range=0.2,            # Zoom in/out
    horizontal_flip=True       # Flip horizontally
)
```

IMAGES AFTER AUGMENTATION:

IMPLEMENTATION DETAILS

- MODEL TRAINING:
  - Labels 0 = not fractured; 1 = fractured
- CNN: 4 Layers (Conv2D, MaxPooling, 1 Dropout Layer, Flatten, Dense; activation: relu & sigmoid)

IMPLEMENTATION DETAILS

- SETTING UP AN APPLICATION WITH FLASK:
  - Set up a venv through the terminal
  - Install & import Flask
  - Create an instance of the Flask-class
  - Use route() decorator to set URL
  - Save file
  - Run app through terminal

```
from flask import Flask

app = Flask(__name__)
```

```
(base)                                  ~ % mkdir myproject
(base)                                  ~ % cd myproject
(base)                                  myproject % python3 -m venv .venv
(base)                                  myproject % . .venv/bin/activate
(.venv)                                         myproject % pip3 install Flask
Collecting Flask
  Downloading flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug>=3.0.0 (from Flask)
  Using cached werkzeug-3.0.4-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.1.2 (from Flask)
  Using cached jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.1.2 (from Flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from Flask)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
  Downloading blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->Flask)
  Using cached MarkupSafe-3.0.1-cp312-cp312-macosx_11_0_arm64.whl.metadata (4.0 kB)
Downloading flask-3.0.3-py3-none-any.whl (101 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 4.0 MB/s eta 0:00:00
Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
Downloading click-8.1.7-py3-none-any.whl (97 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 5.0 MB/s eta 0:00:00
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Using cached jinja2-3.1.4-py3-none-any.whl (133 kB)
Using cached werkzeug-3.0.4-py3-none-any.whl (227 kB)
Using cached MarkupSafe-3.0.1-cp312-cp312-macosx_11_0_arm64.whl (12 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
Successfully installed Flask-3.0.3 Jinja2-3.1.4 MarkupSafe-3.0.1 Werkzeug-3.0.4 blinker-1.8.2 click-
8.1.7 itsdangerous-2.2.0
```

IMPLEMENTATION DETAILS

- MODEL EVALUATION:
  - Accuracy ranged from **93.75 % - 96.88 %** (ran on AWS Sagemaker)

```
288/288 ━━━━━━━━━━━━━━━━━━━━ 134s 466ms/step - accuracy: 0.9260 - loss: 0.2004 - val_accuracy: 0.9745 - val_loss: 0.0748
Epoch 18/20
288/288 ━━━━━━━━━━━━━━━━━━━━ 1s 316us/step - accuracy: 0.9375 - loss: 0.1495 - val_accuracy: 1.0000 - val_loss: 0.0159
Epoch 19/20
288/288 ━━━━━━━━━━━━━━━━━━━━ 132s 457ms/step - accuracy: 0.9397 - loss: 0.1715 - val_accuracy: 0.9770 - val_loss: 0.0689
Epoch 20/20
288/288 ━━━━━━━━━━━━━━━━━━━━ 1s 315us/step - accuracy: 0.9375 - loss: 0.1073 - val_accuracy: 1.0000 - val_loss: 0.0192
```

THANK YOU!

IMAGE PROCESSING& COMPUTER VISION: **FRACTURED BONES**

by Julia Nuss