Bone Fracture Detection via X-Ray Image Processing

Zhoubin Maneshi

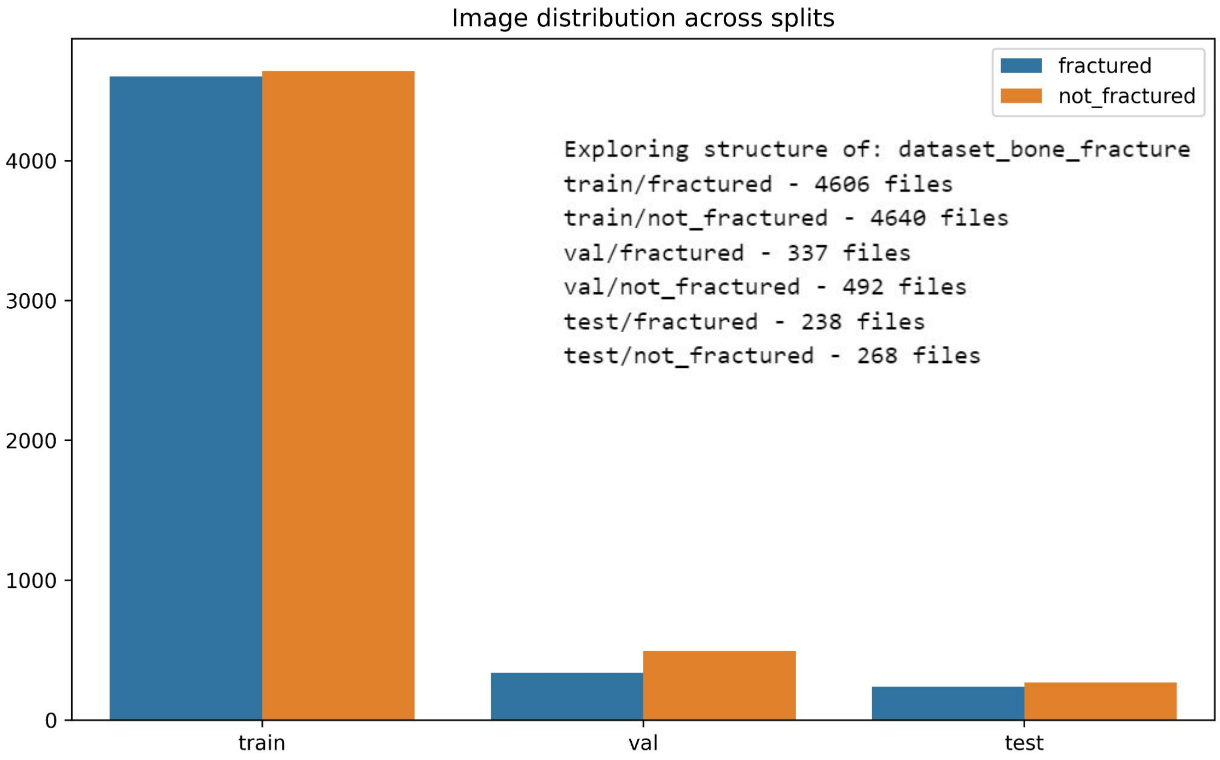
11.04.2025

Overview

- **Dataset**: a collection of radiographic images (`IMG_SIZE=(224,224)`, in greyscale) covering all anatomical body regions (limbs, hips, knees, ...)
- **Categories:** train, test, and validation
- **Subcategories**: fractured and non-fractured
- **Challenges:**
  - Dataset contains corrupted images (removed via cleaning)
  - Random file names (unimportant when using **`ImageDataGenerator`**)
  - Multiple copies of the same image
  - Rotated images

Sample Image Grid

Image Distribution across Splits



Preprocessing and Augmentation

- **Constants:** `IMG_SIZE=(224,224), BATCH_SIZE=32`
- **Random Transformations:** rotation, shift, zoom, flips
- **Normalization:** rescales pixel values to `[0,1]` for better Neural Network performance
- **`ImageDataGenerator`** is applied for training, validation, and testing a binary classification task
  - **`shuffle=False`** for test data ensures prediction
  - Class labels are inferred from subfolder names (frac, non-frac)

Build a Model

- **Architecture:** sequential
- **Convolutional Layers:** 32 filters, ReLU activation, `3x3` Kernel size, `224x224x3` input shape
- **Pooling Layers:** `MaxPooling(2,2)` downsamples feature maps
- **Flatenning**: to create 1D dense layers
- **Dropout (0.5)** drops 50% of neurons during training
- **Dense** sets up fully connected layers
- **Final Layer:** 1 neuron (binary classification) and sigmoid activation

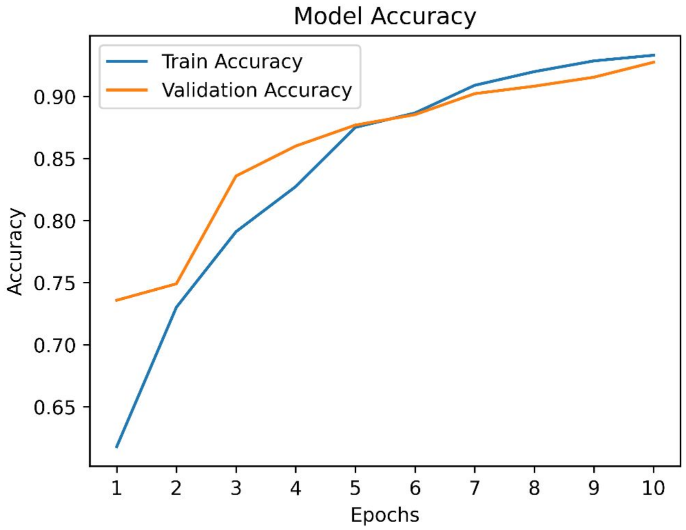
Model Summary

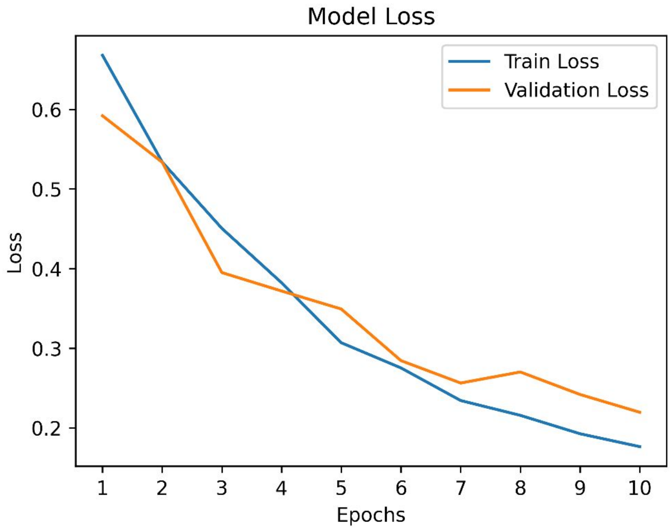
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 max_pooling2d (MaxPooling2  (None, 111, 111, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 54, 54, 64)        0
 g2D)

 flatten (Flatten)           (None, 186624)            0

 dropout (Dropout)           (None, 186624)            0

 dense (Dense)               (None, 128)               23888000

 dense_1 (Dense)             (None, 1)                 129
```

Training Phase

```
289/289 [==============================] - 631s 2s/step - loss: 0.6680 - accuracy: 0.6179 - val_loss: 0.5920 - val_accuracy: 0.7358
Epoch 2/10
289/289 [==============================] - 486s 2s/step - loss: 0.5340 - accuracy: 0.7302 - val_loss: 0.5336 - val_accuracy: 0.7491
Epoch 3/10
289/289 [==============================] - 477s 2s/step - loss: 0.4506 - accuracy: 0.7911 - val_loss: 0.3950 - val_accuracy: 0.8359
Epoch 4/10
289/289 [==============================] - 424s 1s/step - loss: 0.3823 - accuracy: 0.8274 - val_loss: 0.3719 - val_accuracy: 0.8601
Epoch 5/10
289/289 [==============================] - 380s 1s/step - loss: 0.3069 - accuracy: 0.8751 - val_loss: 0.3493 - val_accuracy: 0.8770
Epoch 6/10
289/289 [==============================] - 399s 1s/step - loss: 0.2753 - accuracy: 0.8867 - val_loss: 0.2845 - val_accuracy: 0.8854
Epoch 7/10
289/289 [==============================] - 401s 1s/step - loss: 0.2343 - accuracy: 0.9090 - val_loss: 0.2563 - val_accuracy: 0.9023
Epoch 8/10
289/289 [==============================] - 433s 1s/step - loss: 0.2157 - accuracy: 0.9200 - val_loss: 0.2702 - val_accuracy: 0.9083
Epoch 9/10
289/289 [==============================] - 414s 1s/step - loss: 0.1926 - accuracy: 0.9287 - val_loss: 0.2419 - val_accuracy: 0.9156
Epoch 10/10
289/289 [==============================] - 398s 1s/step - loss: 0.1764 - accuracy: 0.9332 - val_loss: 0.2196 - val_accuracy: 0.9276
```

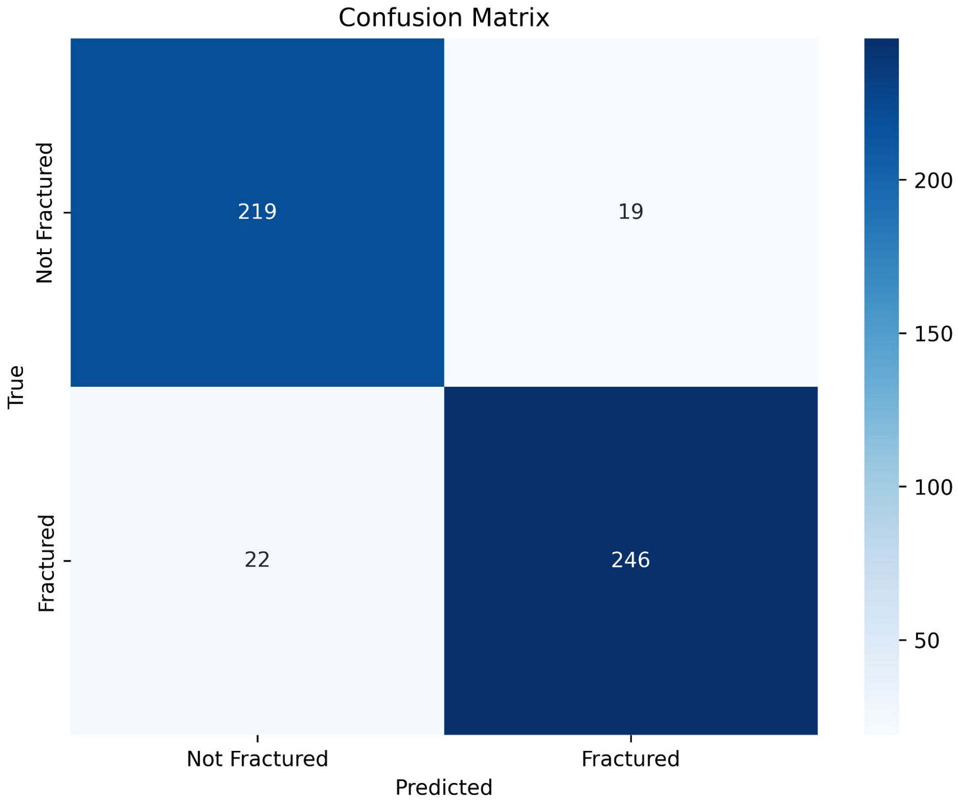
Accuracy and Loss






```
16/16 [==============================] - 4s 248ms/step - loss: 0.2476 - accuracy: 0.9190
Test Accuracy: 91.90%
16/16 [==============================] - 4s 214ms/step
               precision    recall  f1-score   support

not_fractured       0.91      0.92      0.91       238
    fractured       0.93      0.92      0.92       268

     accuracy                           0.92       506
    macro avg       0.92      0.92      0.92       506
 weighted avg       0.92      0.92      0.92       506
```

Confusion Matrix



Further Improvement

- **Use of Hashing to remove duplicates**
  - Avoids data leakage, class imbalance, and wasted computation
- **Use of Transfer Learning**
  - MobileNetV2 is lightweight and efficient using a pre-trained model for better performance of medical imaging
- **Rotation Invariance Strategies**
  - Add rotation invariance strategies or augmentation tricks for rotated duplicates
- **Wise choice of filename** for better reproducibility, debugging, and interpretation

Thank you!