Fracture Detection in Musculoskeletal Radiographs

*

1st Alexandra-Camelia Ștefan
Faculty of Automatic Control and Computer Science
National University Politehnica of Bucharest
Bucharest, Romania
alexandrastefan911@gmail.com

2th Maria-Andreea Tudosie
Faculty of Automatic Control and Computer Science
National University Politehnica of Bucharest
Bucharest, Romania
mariatudosie53@gmail.com

I. Introduction

This document is a report on the second milestone of the Fracture Detection project. The purpose of this part of the project is to obtain the required dataset, preprocess the data and implement a baseline model. This paper will describe the process of the data preparation and the results of the proposed model.

A. Motivation for choosing MURA-v1.1

The MURA-v1.1 (Musculoskeletal Radiographs) dataset was selected for this fracture detection project due to its status as one of the largest publicly available datasets designed specifically for musculoskeletal radiograph analysis. It contains radiographic images, annotated by expert radiologists, covering a wide range of anatomical regions, including the wrist, elbow, shoulder, and fingers. This comprehensive and diverse dataset provides a robust foundation for training and evaluating models. Additionally, the high-quality, expert-labeled annotations in MURA-v1.1 enhance the credibility of the supervised learning process by providing accurate ground truth data. Its use aligns with the project's aim to address clinically relevant challenges in musculoskeletal imaging and allows for benchmarking against other studies in the field. This makes MURA-v1.1 an ideal choice for advancing automated fracture detection research.

II. MURA-v1.1 Dataset Description

MURA-v1.1 (Musculoskeletal Radiographs Dataset) is a large-scale dataset developed for the analysis and classification of musculoskeletal abnormalities in radiographs. It is one of the largest public datasets of its kind and is widely used in medical imaging research, particularly for tasks involving fracture detection and abnormality classification.

Size and Scope: MURA-v1.1 consists of 40,561 labeled images from 14,863 patient studies, making it a comprehensive dataset for musculoskeletal radiographic analysis.

Anatomical Coverage: The dataset includes radiographs from seven anatomical regions:

- Elbow
- Finger
- Hand
- Humerus
- Forearm
- Shoulder
- Wrist

Labels: Each study (a collection of one or more radiographic images of a single anatomical region) is labeled as either "Normal" or "Abnormal", based on the findings of expert radiologists. These labels serve as the ground truth for training and evaluation purposes.

Clinical Relevance: MURA is designed to simulate real-world clinical workflows. The abnormalities detected in the dataset include fractures, lesions, and other musculoskeletal pathologies, reflecting the variety and complexity seen in clinical practice.

Diversity: The dataset includes a broad age range and varying radiographic projections, offering a diverse sample to ensure generality in machine learning models.

Annotations: Annotations were provided by board-certified radiologists, ensuring high reliability and clinical accuracy. This quality is critical for supervised learning tasks requiring precise labels.

Format: Images are stored in both grayscale and RGB format, with a resolution adequate for use in deep learning applications. The dataset is organized by patient studies, facilitating easier access and segmentation for training and testing purposes.

Applications: MURA-v1.1 is widely used for training and benchmarking deep learning models for abnormality detection, binary classification of radiographs, segmentation, and other diagnostic imaging tasks. Its size, diversity, and clinical alignment make it a gold standard for musculoskeletal imaging research.

A. Preprocessing data

Color properties: To evaluate the color properties of images in the dataset, the images were classified based on their channel modes. Specifically, the code iterates through all files in a specified directory, analyzing each image to determine whether it is in RGB (three channels) or grayscale (single channel, "L" mode).

Results:

The results of the analysis indicated the presence of both RGB images and grayscale images. Identifying the proportions

of RGB and grayscale images allows for appropriate conversion or normalization to ensure uniformity across the dataset.

```
Checking the properties of images in the traininig directory:
----->There are 14631 RGB images and 22177 grayscale images in the directory

Checking the properties of images in the testing directory:
----->There are 1358 RGB images and 1839 grayscale images in the directory

(1358, 1839)
```

Fig. 1. Results of color properties check

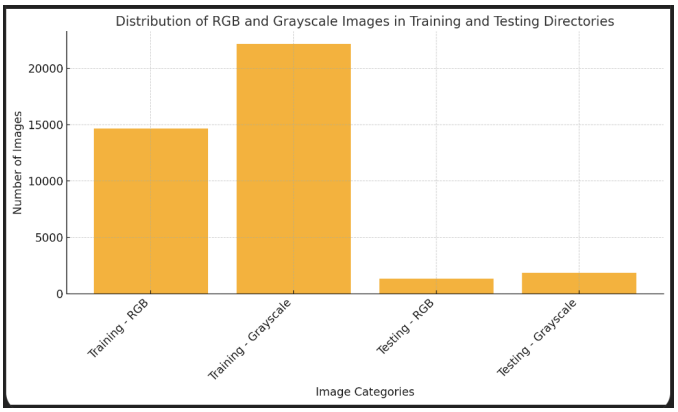

Fig. 2. Visualization of channel mode diversity

B. Image Dimension Analysis

The get_image_sizes(directory) function was developed to analyze the dimensions of images in a specified directory. It traverses all image files within the directory, records their width and height, and determines whether the dataset contains images with consistent dimensions. This functionality is critical in image preprocessing workflows, as uniform dimensions are often required for training models. Identifying inconsistencies allows for necessary resizing or padding to ensure compatibility and improve model performance.

Results:

```
Checking the sizes of images in the training directory:
Images have different dimentions. Widths varry between 89 and 512 pixels.
Heights varry between 132 and 512 pixels.

Checking the sizes of images in the testing directory:
Images have different dimentions. Widths varry between 108 and 512 pixels.
Heights varry between 151 and 512 pixels.
```

Fig. 3. Existent dimension formats in dataset

Fig. 4. Visualization of image size variety

C. Image Pixel Intensity Scale Analysis

Another step in prepocessing the data is to evaluate the consistency of pixel intensity scales across all images in a specified directory. This analysis is essential because variations in pixel intensity scales can disrupt normalization and standardization processes during preprocessing, leading to inconsistencies in model performance. Ensuring a uniform scale across the data set facilitates seamless pre-processing.

Results:

```
Inconsistent scale found in image: image1.png
Expected range: [0, 255], but got [1, 183]
For training:
Images are on the same scale: False
Interval of pixels values: (0, 255)
Inconsistent scale found in image: image3.png
Expected range: [0, 255], but got [0, 250]

For testing:
Images are on the same scale: False
Interval of pixels values: (0, 255)
```

Fig. 5. Pixel intensity statistics

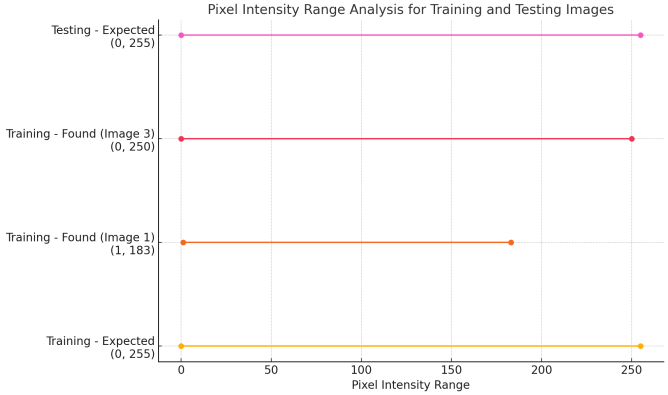

Fig. 6. Pixel intensity visualization

III. BoneFractureDataset: Custom Dataset Implementation

The BoneFractureDataset class is a custom implementation derived from PyTorch's Dataset class, designed to handle the loading and preprocessing of a labeled dataset of bone fracture images. This class simplifies the process of managing image paths, labels, and transformations.

A. Key Features

- **Initialization (__init__):**
  The class accepts two CSV files as inputs:
  - `labeled_train_file`: Contains the directory names and their associated labels.
  - `paths_file`: Contains the full paths to the images.

  It constructs a mapping (`label_dict`) that links directories to their respective labels using the data from the `labeled_train_file`. The class iterates through the image paths in `paths_file` and associates each image with its corresponding label based on its parent directory.
- **Data Storage:**
  The dataset stores a list of tuples, each containing:
  - An image path.
  - Its corresponding label.

  This structure enables efficient indexing and retrieval.
- **Length (__len__):**
  The method returns the total number of images in the dataset, making it compatible with PyTorch's data handling utilities.
- **Item Retrieval (__getitem__):**
  This method retrieves an image, its label, and its file path based on the specified index. Images are loaded using the PIL library and converted to RGB format to ensure consistency.
- **Optional transformations** (e.g., resizing, normalization) can be applied to the images, allowing flexibility in preprocessing.
- **Significance:**
  This class provides a streamlined way to handle datasets with complex directory structures and labeling systems. By abstracting the loading, labeling, and transformation processes, it ensures seamless integration with PyTorch's `DataLoader` for batch processing during training and evaluation. Additionally, its extensibility allows for easy customization to accommodate different preprocessing needs or dataset structures.

B. The Datasets Sizes

- Training images: 29446
- Validation images: 7362
- Testing images: 319

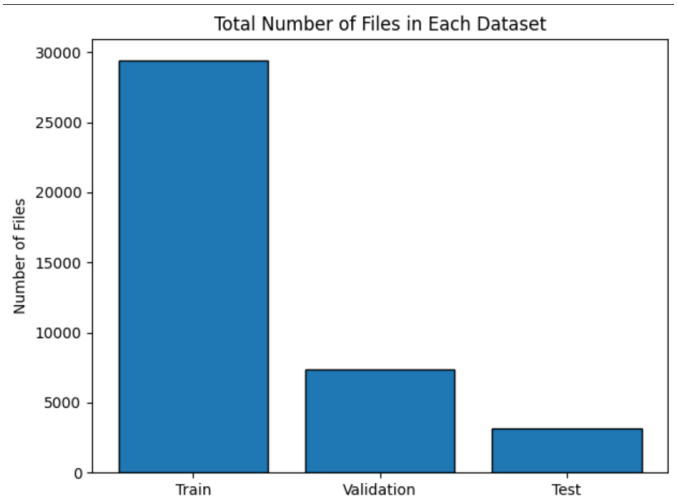

Fig. 7. The sizes of the Training, Validation and Testing Datasets

C. Class Distribution in our datasets

Train Dataset distribution

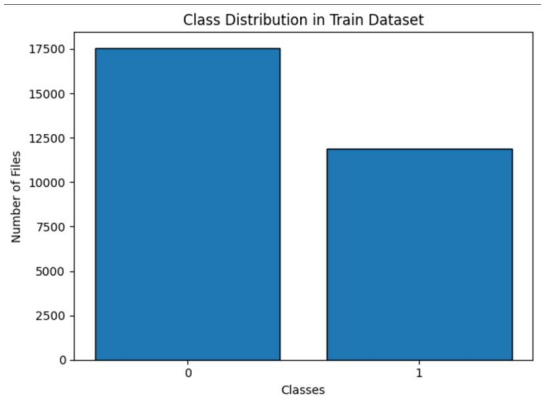

Fig. 8. Class Distribution in Train Dataset

Validation Dataset distribution

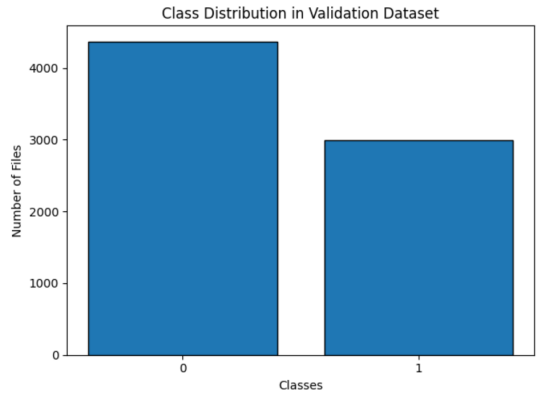

Fig. 9. Class Distribution in Validation Dataset

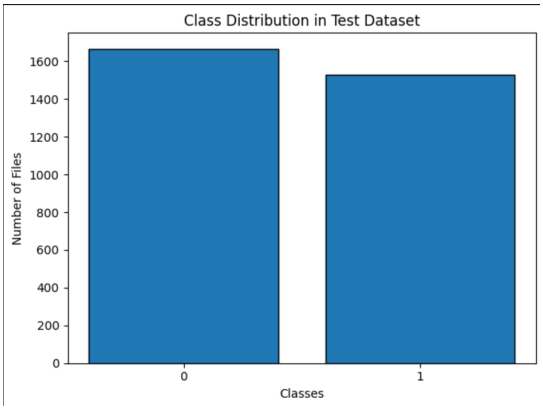

Fig. 10. Class Distribution in Test Dataset

IV. The Model Training

A. Resnet50

The model learns steadily on the training set, as evidenced by the declining training loss and increasing accuracy. However, the gap between training and validation metrics, along with fluctuating validation loss, suggests overfitting. The poor performance on the test set reflects the model's inability to generalize well, which is a common issue when training from scratch without leveraging pretrained weights. The non-pretrained ResNet-50 model demonstrated moderate performance in classifying bone fractures, with steady improvements in training accuracy but limited generalization to the validation and test sets. The training loss decreased consistently over the 25 epochs, starting from 0.6944 in the first epoch and reaching 0.4699 by the final epoch, reflecting the model's ability to learn from the training data. The training accuracy also showed a gradual improvement, starting at 58.74% and reaching 77.87% by the 25th epoch. While these results indicate that the non-pretrained model is capable of fitting the training data, its overall performance was significantly slower and less effective compared to the pretrained counterpart, highlighting the challenges of training such a model from scratch without leveraging previously learned features.

```
Epoch [1/25] - Train Loss: 0.6944, Train Accuracy: 0.5874 - Val Loss: 0.6830, Val Accuracy: 0.5954
Epoch [2/25] - Train Loss: 0.6639, Train Accuracy: 0.5992 - Val Loss: 0.6791, Val Accuracy: 0.5922
Epoch [3/25] - Train Loss: 0.6564, Train Accuracy: 0.6056 - Val Loss: 0.6575, Val Accuracy: 0.6038
Epoch [4/25] - Train Loss: 0.6492, Train Accuracy: 0.6124 - Val Loss: 0.6608, Val Accuracy: 0.5941
Epoch [5/25] - Train Loss: 0.6508, Train Accuracy: 0.6133 - Val Loss: 0.6515, Val Accuracy: 0.6091
Epoch [6/25] - Train Loss: 0.6450, Train Accuracy: 0.6150 - Val Loss: 0.6659, Val Accuracy: 0.5899
Epoch [7/25] - Train Loss: 0.6368, Train Accuracy: 0.6269 - Val Loss: 0.6324, Val Accuracy: 0.6406
Epoch [8/25] - Train Loss: 0.6289, Train Accuracy: 0.6367 - Val Loss: 0.6338, Val Accuracy: 0.6475
Epoch [9/25] - Train Loss: 0.6165, Train Accuracy: 0.6547 - Val Loss: 0.6608, Val Accuracy: 0.6301
Epoch [10/25] - Train Loss: 0.5987, Train Accuracy: 0.6696 - Val Loss: 0.7590, Val Accuracy: 0.6159
Epoch [11/25] - Train Loss: 0.5856, Train Accuracy: 0.6870 - Val Loss: 0.6075, Val Accuracy: 0.6838
Epoch [12/25] - Train Loss: 0.5717, Train Accuracy: 0.7003 - Val Loss: 0.6225, Val Accuracy: 0.6785
Epoch [13/25] - Train Loss: 0.5637, Train Accuracy: 0.7100 - Val Loss: 0.5551, Val Accuracy: 0.7190
Epoch [14/25] - Train Loss: 0.5562, Train Accuracy: 0.7143 - Val Loss: 0.6066, Val Accuracy: 0.6883
Epoch [15/25] - Train Loss: 0.5519, Train Accuracy: 0.7184 - Val Loss: 0.6188, Val Accuracy: 0.6680
Epoch [16/25] - Train Loss: 0.5385, Train Accuracy: 0.7300 - Val Loss: 0.5701, Val Accuracy: 0.7160
Epoch [17/25] - Train Loss: 0.5354, Train Accuracy: 0.7334 - Val Loss: 0.5586, Val Accuracy: 0.7180
Epoch [18/25] - Train Loss: 0.5250, Train Accuracy: 0.7387 - Val Loss: 0.7088, Val Accuracy: 0.6024
Epoch [19/25] - Train Loss: 0.5265, Train Accuracy: 0.7391 - Val Loss: 0.6389, Val Accuracy: 0.6841
Epoch [20/25] - Train Loss: 0.5175, Train Accuracy: 0.7459 - Val Loss: 0.5460, Val Accuracy: 0.7270
Epoch [21/25] - Train Loss: 0.5071, Train Accuracy: 0.7512 - Val Loss: 0.5763, Val Accuracy: 0.7209
Epoch [22/25] - Train Loss: 0.4986, Train Accuracy: 0.7584 - Val Loss: 0.5411, Val Accuracy: 0.7297
Epoch [23/25] - Train Loss: 0.4895, Train Accuracy: 0.7650 - Val Loss: 0.5494, Val Accuracy: 0.7179
Epoch [24/25] - Train Loss: 0.4802, Train Accuracy: 0.7702 - Val Loss: 0.6012, Val Accuracy: 0.6964
Epoch [25/25] - Train Loss: 0.4699, Train Accuracy: 0.7787 - Val Loss: 0.9051, Val Accuracy: 0.5470
Test Accuracy: 0.5790
Test F1-Score: 0.6582
Test ROC-AUC: 0.6686
```

Fig. 11. Resnet50 training

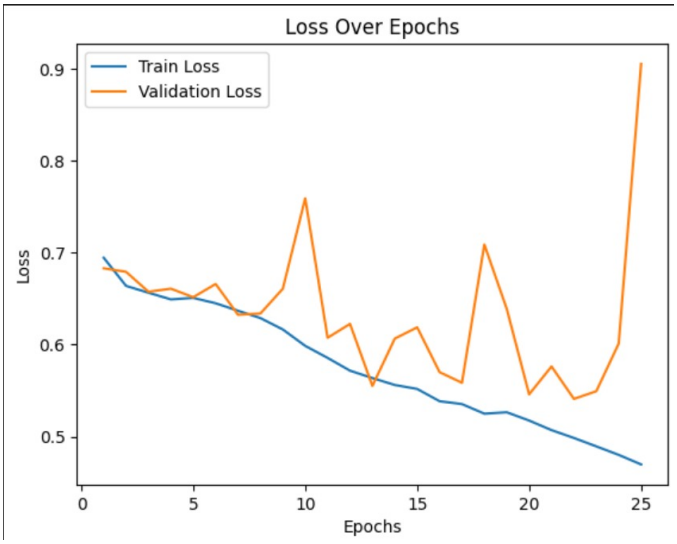

Fig. 12. Resnet50 training and validation loss

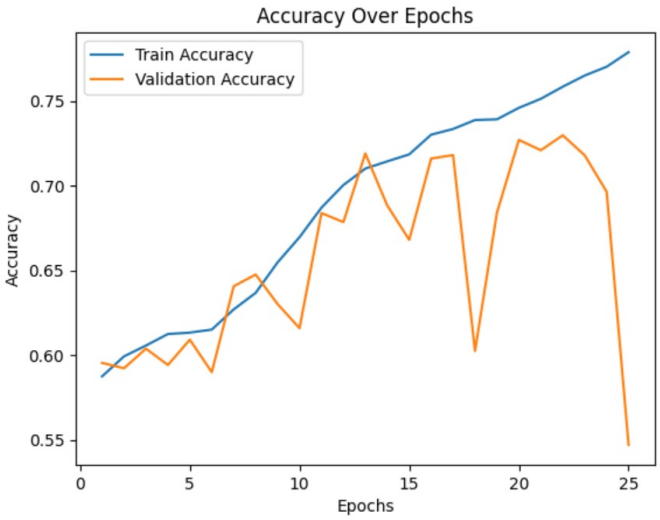

Fig. 13. Resnet50 training and validation Accuracy

The model learns steadily on the training set, as evidenced by the declining training loss and increasing accuracy. However, the gap between training and validation metrics, along with fluctuating validation loss, suggests overfitting.

The confusion matrix indicates imbalanced predictions, with a higher focus on correctly identifying fractured bones (TP) at the expense of misclassifying non-fractured samples (FP).

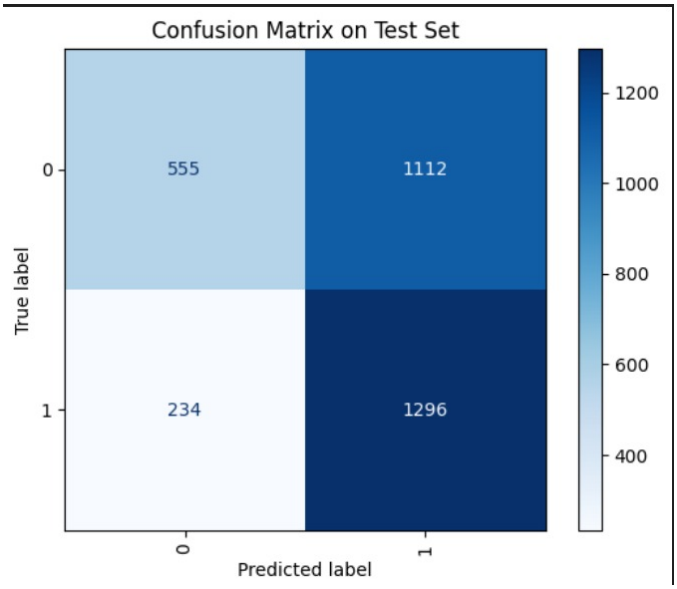

Fig. 14. Resnet50 Confussion Matrix

B. Pretrained Resnet50

The pretrained ResNet-50 model demonstrated strong performance in classifying bone fractures, with high accuracy on the training set and reasonable validation and test accuracy.The training loss decreased steadily across the 25 epochs, starting from 0.5383 in the first epoch and reaching 0.0398 by the 25th epoch, indicating that the model was learning effectively from the data. The training accuracy improves rapidly, reaching 98.50% by the end of the 25th epoch. This high accuracy demonstrates that the pretrained model successfully fits the training data.

```
Epoch [1/25] - Train Loss: 0.5383, Train Accuracy: 0.7404 - Val Loss: 0.5910, Val Accuracy: 0.7187
Epoch [2/25] - Train Loss: 0.4896, Train Accuracy: 0.7709 - Val Loss: 0.6907, Val Accuracy: 0.7335
Epoch [3/25] - Train Loss: 0.4627, Train Accuracy: 0.7908 - Val Loss: 0.5892, Val Accuracy: 0.7180
Epoch [4/25] - Train Loss: 0.4390, Train Accuracy: 0.8039 - Val Loss: 0.4853, Val Accuracy: 0.7756
Epoch [5/25] - Train Loss: 0.4200, Train Accuracy: 0.8154 - Val Loss: 0.5403, Val Accuracy: 0.7596
Epoch [6/25] - Train Loss: 0.3883, Train Accuracy: 0.8337 - Val Loss: 0.7579, Val Accuracy: 0.7032
Epoch [7/25] - Train Loss: 0.3562, Train Accuracy: 0.8468 - Val Loss: 0.5332, Val Accuracy: 0.7688
Epoch [8/25] - Train Loss: 0.3115, Train Accuracy: 0.8698 - Val Loss: 0.5068, Val Accuracy: 0.7914
Epoch [9/25] - Train Loss: 0.2577, Train Accuracy: 0.8941 - Val Loss: 0.5996, Val Accuracy: 0.7757
Epoch [10/25] - Train Loss: 0.2125, Train Accuracy: 0.9109 - Val Loss: 0.6404, Val Accuracy: 0.7696
Epoch [11/25] - Train Loss: 0.1578, Train Accuracy: 0.9377 - Val Loss: 0.9046, Val Accuracy: 0.7566
Epoch [12/25] - Train Loss: 0.1295, Train Accuracy: 0.9499 - Val Loss: 0.7923, Val Accuracy: 0.7521
Epoch [13/25] - Train Loss: 0.1024, Train Accuracy: 0.9610 - Val Loss: 0.9312, Val Accuracy: 0.7683
Epoch [14/25] - Train Loss: 0.0917, Train Accuracy: 0.9650 - Val Loss: 0.9634, Val Accuracy: 0.7722
Epoch [15/25] - Train Loss: 0.0866, Train Accuracy: 0.9679 - Val Loss: 1.0208, Val Accuracy: 0.7323
Epoch [16/25] - Train Loss: 0.0681, Train Accuracy: 0.9747 - Val Loss: 0.9469, Val Accuracy: 0.7786
Epoch [17/25] - Train Loss: 0.0665, Train Accuracy: 0.9757 - Val Loss: 1.0635, Val Accuracy: 0.7616
Epoch [18/25] - Train Loss: 0.0689, Train Accuracy: 0.9740 - Val Loss: 1.0218, Val Accuracy: 0.7620
Epoch [19/25] - Train Loss: 0.0495, Train Accuracy: 0.9816 - Val Loss: 1.0465, Val Accuracy: 0.7630
Epoch [20/25] - Train Loss: 0.0631, Train Accuracy: 0.9763 - Val Loss: 1.0041, Val Accuracy: 0.7721
Epoch [21/25] - Train Loss: 0.0426, Train Accuracy: 0.9854 - Val Loss: 1.2066, Val Accuracy: 0.7711
Epoch [22/25] - Train Loss: 0.0634, Train Accuracy: 0.9768 - Val Loss: 1.1051, Val Accuracy: 0.7592
Epoch [23/25] - Train Loss: 0.0473, Train Accuracy: 0.9823 - Val Loss: 1.2469, Val Accuracy: 0.7571
Epoch [24/25] - Train Loss: 0.0397, Train Accuracy: 0.9860 - Val Loss: 1.1835, Val Accuracy: 0.7795
Epoch [25/25] - Train Loss: 0.0398, Train Accuracy: 0.9850 - Val Loss: 1.2480, Val Accuracy: 0.7702
Test Accuracy: 0.7335
Test F1-Score: 0.7048
Test ROC-AUC: 0.7974
```

Fig. 15. Pretrained Resnet50 training

Test Set Performance

- Test Accuracy of 73.35% indicates the model's overall correctness on the test data.
- Test F1-Score of 70.48% indicates moderate success in identifying both classes (fractured and non-fractured bones).
- Test ROC-AUC of 79.74% reflects the model's ability to distinguish between the two classes across different thresholds.

However, the validation loss showed fluctuations, particularly after the initial epochs. This discrepancy suggests that while the model fits the training data well, it struggles to generalize to unseen data in the validation set.

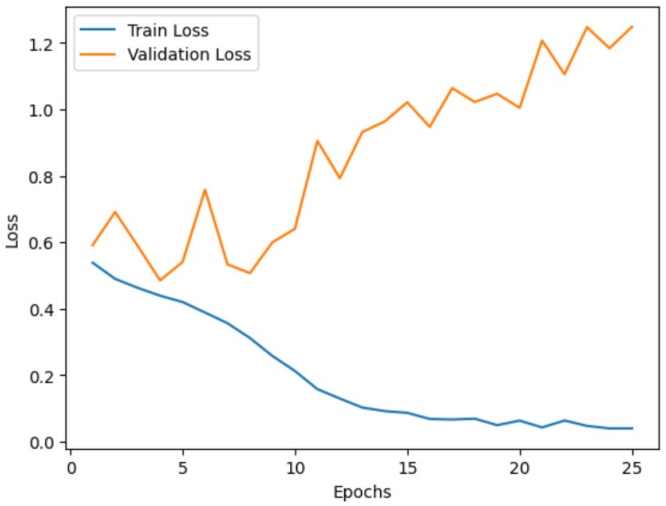

Fig. 16. Pretrained Resnet50 training and validation loss

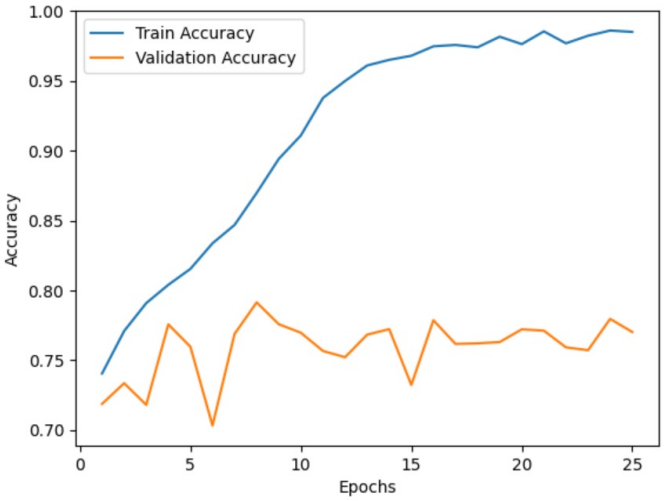

Fig. 17. Pretrained Resnet50 training and validation accuracy

- Test Accuracy of 57.90% suggests that the model performs poorly on the test set, indicating limited generalization and difficulty in distinguishing between the two classes effectively.
- The F1-Score of 65.82% reflects a balance between precision and recall but remains low, emphasizing the model's struggles in correctly identifying both fractured and non-fractured samples. ROC-AUC: 66.86
- The ROC-AUC score of 66.86% highlights the model's ability to distinguish between the two classes across

different thresholds. While acceptable, this value is far from ideal.

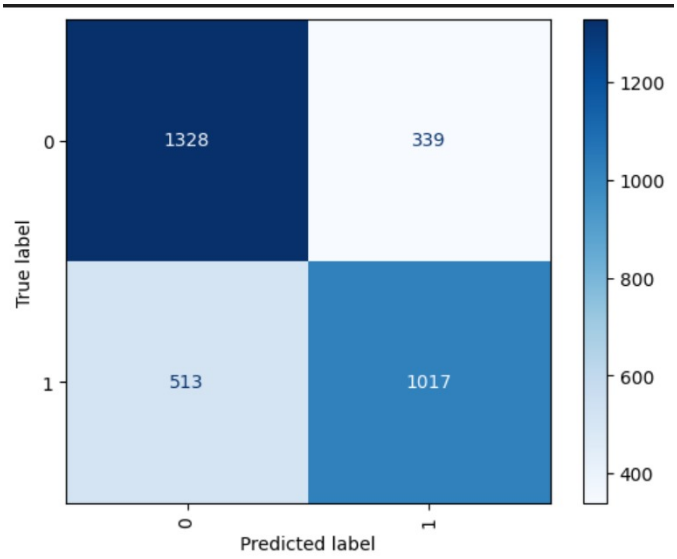

Fig. 18. Pretrained Resnet50 Confussion Matrix

C. Resnet50 vs. Pretrained Resnet50

The pretrained model achieved higher accuracy on both the training and validation sets. It converged much faster, leveraging the existing feature knowledge from ImageNet. In contrast, the non-pretrained model exhibited slower learning and required more epochs to achieve a comparable training accuracy. However, its validation accuracy fluctuated significantly, indicating challenges in generalization. The pretrained model produced better classification results, with fewer false positives and false negatives. Its confusion matrix showed a higher count of true positives and true negatives, reflecting improved ability to distinguish between fractured and non-fractured bones. On the other hand, the non-pretrained model struggled with a large number of misclassifications, highlighting its weaker understanding of the task.

REFERENCES

[1] Pranav Rajpurkar, Jeremy Irvin, Aarti Bagul, Daisy Ding, Tony Duan, Hershel Mehta, Brandon Yang, Kaylie Zhu, Dillon Laird, Robyn L. Ball, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, Andrew Y. Ng "MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs,"

[2] Dataset download, https://www.kaggle.com/datasets/anishnaskar/mura-version-11.