National University of Computer and Emerging Sciences, Lahore

Data Mining

Project Report

Bone Fracture Detection

Student Name	Roll No.
1. Fayez Ali	21l-6228
2. Jazib Ali	21l-6236
3. Zahra Hussain	21l-5615

12 May 2024

1 Introduction

Bone fractures are common injuries that require accurate and timely diagnosis for effective treatment. Traditional methods of diagnosing fractures, such as X-rays, can be time-consuming and may require specialized equipment. In this project, we aim to develop a deep learning-based system for bone fracture detection using image data. By leveraging machine learning algorithms, we can automate the process of fracture detection, leading to quicker diagnosis and treatment.

1.1 Motivation

The motivation behind this project stems from the need for a faster and more accessible method of diagnosing bone fractures. By utilizing image data and deep learning techniques, we can create a system that can accurately detect fractures from medical images, such as X-rays or MRI scans. This can significantly reduce the time required for diagnosis and improve patient outcomes by enabling prompt treatment.

1.2 Problem Scope

The problem we are addressing is binary classification: given an input medical image, classify whether the bone is fractured or not. This binary classification problem simplifies the complex task of diagnosing fractures into a machine learning task that can be tackled using supervised learning algorithms.

2 Data set Description

We obtained the image dataset from Kaggle, which consists of X-ray images of bones with and without fractures. It is categorized into two classes: fractured and non-fractured. Each image represents a single X-ray scan of a bone, captured using medical imaging equipment. The dataset was preprocessed by resizing the images to 224x224 pixels, a common size for deep learning models such as CNNs and transformers. We used techniques such as data augmentation to increase the diversity of the training data and improve model generalization.

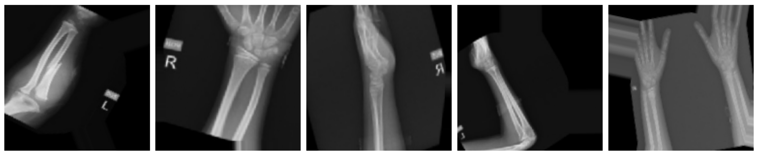

Figure 1: Dataset Visualization

2.1 Data Preprocessing

In the preprocessing stage of our bone fracture detection project, we applied resizing and data augmentation techniques were applied to augment the dataset and increase its diversity. Data augmentation involves applying a variety of transformations to the images, such as rotation, flipping, and zooming, to create new variations of the original data samples. By introducing these variations, data augmentation helps prevent overfitting and improves the generalization ability of the trained models. For instance, rotating or flipping an image does not change its underlying characteristics but provides additional perspectives for the model to learn from.
Finally, we split the dataset into a training set, used for model training, and a test set, held out for model evaluation. This preprocessing pipeline aimed to optimize the dataset for training deep learning models and facilitate unbiased evaluation of their performance.

3 Binary Classification

In our bone fracture detection project, we conducted experiments with multiple deep learning models to identify the most effective architecture for accurately detecting fractures in X-ray images. We carefully considered four distinct models: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Transfer Learning with EfficientNetB0, and Vision Transformer (ViT). The selection of these models was based on their suitability for image classification tasks and their performance on similar datasets in previous research. MLP served as a baseline, CNN excelled in capturing spatial features, Transfer Learning leveraged pre-trained models, and ViT explored transformer-based architectures. Our analysis aimed to identify the most suitable model for accurately detecting fractures in X-ray images.

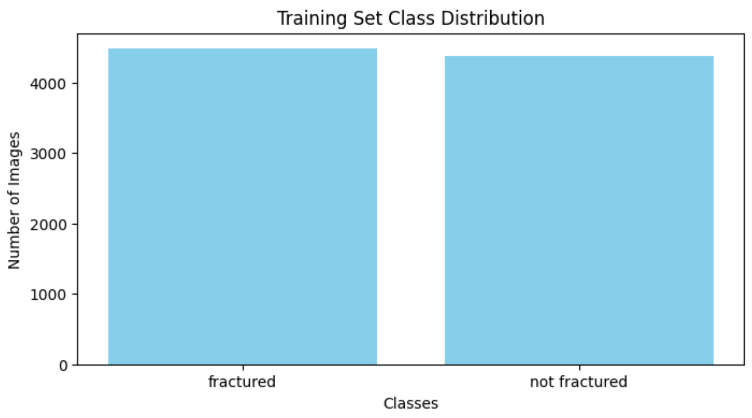

Figure 2: Training Set Class Distribuation

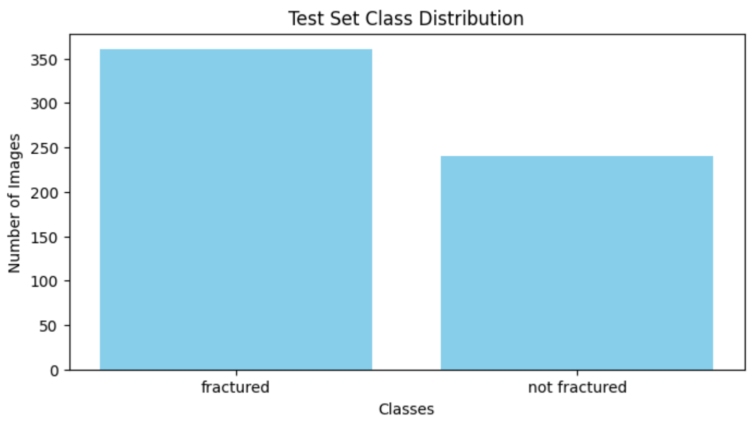

Figure 3: Testing Set Class Distribuation

3.1 Class Imbalance

In our project, one of the primary challenges we encountered was the problem of class imbalance within the dataset. This issue arose due to a significant disparity in the number of samples between the fractured and non-fractured bone classes. To address this imbalance, we implemented corrective measures aimed at ensuring a more equitable distribution of samples across both classes.

Specifically, we utilized oversampling techniques targeted at the minority class, which in this case, was the test set of non fractured bone class. By artificially increasing the number of samples belonging to this class through oversampling, we sought to balance the class distribution within the dataset. This approach enabled the machine learning model to receive adequate exposure to both fractured and non-fractured bone samples during training, mitigating the risk of bias towards the majority class.

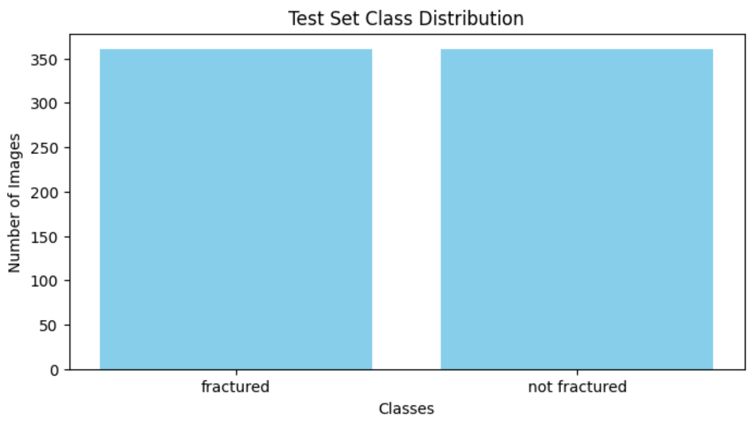

Figure 4: After Oversampling

The implementation of oversampling resulted in a more balanced training process, which was initially 600 , in turn, led to improved model performance. With a more equitable distribution of samples across classes, the model could better learn to distinguish between fractured and non-fractured bones, ultimately enhancing its classification accuracy and overall effectiveness. Moreover, addressing class imbalance helped prevent potential inaccuracies or biases in the model's predictions, ensuring a more reliable bone fracture detection system.

4 MultiLayer Perceptron (MLP)

In our bone fracture detection project, we explored the Multilayer Perceptron (MLP) as one of the deep learning models. Firstly, we define the architecture of our neural network within the Sequential model.

4.1 Architecture:

4.1.1 Input Layer

Our model starts with an input layer tailored to receive images of size 150x150 pixels with three color channels (RGB). These images are then flattened into a 1D array to facilitate processing by subsequent dense layers.

4.1.2 Dense Layer:

We include two dense layers with 512 and 256 neurons, respectively, followed by ReLU activation functions to introduce non-linearity and learn complex patterns within the data.

4.1.3 Dropout Layer:

To prevent overfitting, we incorporate a dropout layer, randomly dropping 50% of input units during training.

4.1.4 Output Layer:

Lastly, a single neuron with a sigmoid activation function serves as the output layer, producing binary classification predictions (fracture or non-fracture).

4.1.5 Model Compilation:

After defining the model architecture, we compile it using Adam optimizer and binary cross-entropy loss function, standard for binary classification. We also monitor accuracy during training. To prevent overfitting, we utilize Early Stopping, halting training if accuracy doesn't improve for a set number of epochs (patience=2). We then restore the weights corresponding to the best accuracy achieved.

4.2 Evaluation of MLP

In this , the accuracy metric for the current epoch is reported as 0.5147, which corresponds to approximately 51.47%. This metric indicates the proportion of correctly classified samples during model training, providing insight into its performance. Additionally, the loss metric for the same epoch is recorded as 3.7756, signifying the magnitude of error between the predicted and actual values.

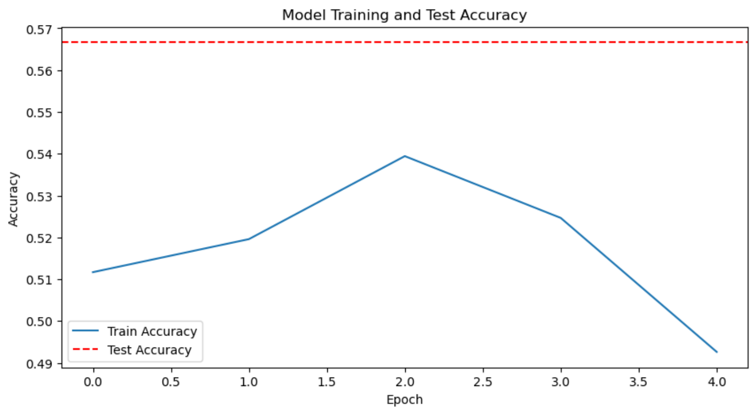

Figure 5: MLP Accuracy

The model's performance metrics are provided:

- **Test Loss:** 0.682
- **Test Accuracy:** 0.566

These values reflect the overall performance of the trained model in detecting bone fractures in X-ray images. The test accuracy of approximately 56.67% indicates the proportion of correctly classified images in the test dataset, while the test loss of approximately 0.6828 represents the average loss incurred during the evaluation process. This evaluation is crucial for assessing the model's effectiveness and determining its readiness for real-world applications in bone fracture detection.

5 Grid Search

The grid search with cross-validation is used to fine-tune the hyperparameters of our custom Keras classifier, which is a neural network model. The hyperparameters include the number of neurons in the hidden layer (`neurons: 256 and 512`), the dropout rate (`dropout_rate: 0.3 and 0.5`), the number of training epochs (`epochs: 10 and 20`), and the batch size (`batch_size: 32 and 64`).

By defining a grid of possible values for these hyperparameters, we systematically explore various combinations. During each iteration of the grid search, the model is trained on a subset of the training data using a specific set of hyperparameters. This training is performed using cross-validation, where the training data is split into multiple folds. The model's performance is evaluated on each fold, and the average performance across all folds is computed.

6 Convolutional Neural Network (CNN)

In our project architecture, we designed a Convolutional Neural Network (CNN) tailored to the specific requirements of our task. The model was built using the TensorFlow framework with Keras API.

6.1 Architecture:

6.1.1 Input Layer:

The input layer was configured to accept images with dimensions of 150x150 pixels and 3 color channels (RGB), ensuring compatibility with our dataset.

6.1.2 Convolutional Layers:

We stacked three convolutional layers, each followed by a rectified linear unit (ReLU) activation function. We set the number of filters in these layers to 32, 64, and 128, respectively, with each filter having a kernel size of 3x3 pixels.

6.1.3 MaxPooling Layers:

Following each convolutional layer, we added a MaxPooling layer to down-sample the feature maps, reducing their spatial dimensions by a factor of 2 in both width and height. This step helped in retaining the most relevant features while reducing computational complexity.

6.1.4 Flatten Layer:

The output of the final convolutional layer was flattened into a one-dimensional array by the Flatten layer. This transformation prepared the feature maps for input into the subsequent fully connected layers.

6.1.5 Fully Connected Layers:

Two dense (fully connected) layers were appended to the architecture, each followed by a ReLU activation function. The first dense layer consisted of 512 neurons, allowing for complex feature mapping and abstraction. The second dense layer contained a single neuron with a sigmoid activation function, making it suitable for binary classification tasks, such as the one we were addressing in our project.

6.1.6 Dropout Layer:

To prevent overfitting and enhance the model's generalization capability, we introduced a Dropout layer with a rate of 0.5 between the two dense layers. This layer randomly dropped out half of the neurons during training, effectively reducing inter-dependencies between neurons and improving the model's robustness.

6.2 Evaluation of CNN

In our training process, we observed the model's performance over 10 epochs. With each epoch, we tracked changes in both accuracy and loss metrics, providing valuable insights into the model's learning dynamics and convergence behavior.

Initially, during the first epoch, the model achieved an accuracy of 54.64% with a corresponding loss of 0.7002. As training progressed, we noticed gradual improvements in both accuracy and loss metrics. By the end of the training phase, the model achieved a remarkable accuracy of 80.12%, accompanied by a significantly reduced loss of 0.4178.This progression suggests that the model successfully extracted and generalized patterns from the training data, thereby enhancing its predictive capabilities.

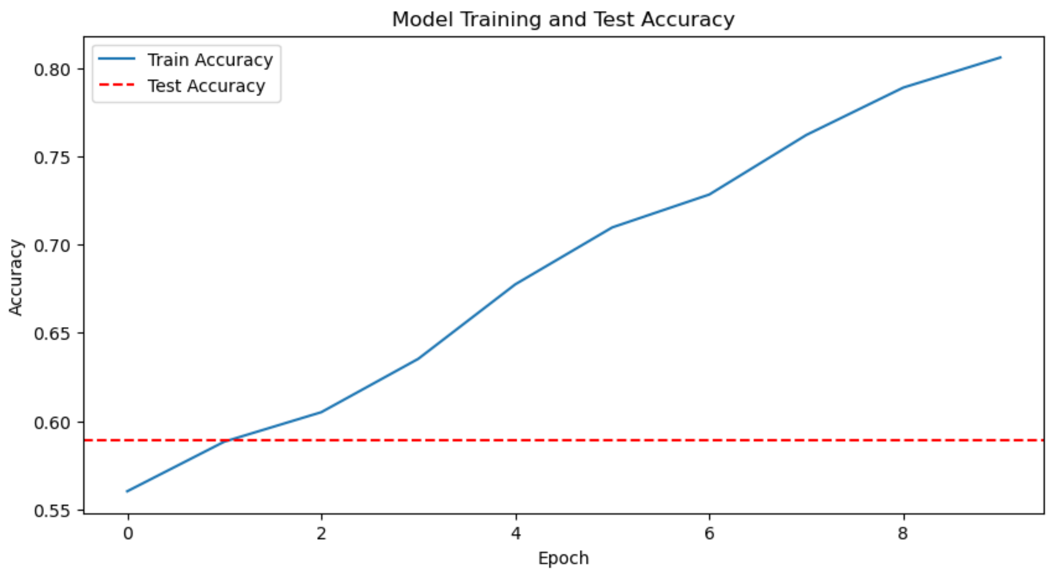

Figure 6: CNN Accuracy

The evaluation process indicates :

- **Test Accuracy:** 0.5888
- **Test Loss:** 0.6439

The test accuracy signifies the proportion of correctly classified samples in the test dataset, implying that the model correctly predicted the class labels for nearly 58.89% of the samples The test loss represents the average loss incurred by the model during predictions on the test dataset. A lower test loss suggests that the model's predictions are closer to the actual labels in the test dataset.

7 Transfer Learning

Transfer learning proves to be a valuable strategy for optimizing model training and achieving superior performance in image classification tasks. It empowers our project by harnessing the knowledge encoded in pre-trained models, addressing data scarcity issues, accelerating training, and enhancing the model's ability to accurately detect bones in medical images.

7.1 Architecture:

7.1.1 Base Model Loading:

The code loads the pre-trained EfficientNetB0 model that is pre-trained on ImageNet data. The `include_top=False` argument excludes the top layers (classification layers) of the model.

7.1.2 Freezing Layers:

Most of the layers in the base model are frozen (non-trainable) except for the last four layers. This freezing prevents the weights in these layers from being updated during training.

7.1.3 Adding Classifier Layers:

New classifier layers are added on top of the base EfficientNetB0 model. First, the output of the base model is flattened to a one-dimensional array. Then, a dense layer with 512 neurons and ReLU activation is added, followed by a dropout layer with a dropout rate of 0.5 to prevent overfitting. Finally, a dense output layer with a single neuron and sigmoid activation is added for binary classification.

7.1.4 Model Compilation:

The model is compiled with the Adam optimizer and binary cross-entropy loss function, suitable for binary classification tasks. The accuracy metric is also specified for evaluation during training.

7.1.5 Training:

The model is trained using the `fit` method with the provided training generator for a specified number of epochs. Additionally, an early stopping callback is employed to monitor the training accuracy and stop training if there is no improvement after a certain number of epochs.

7.2 Evaluation of Transfer Learning

In the training phase, the accuracy fluctuated around the 0.5 across multiple epochs, indicating that the model's performance was no better than random guessing. Despite the extensive training time accuracy remained consistent and the inclusion of ten epochs, the model failed to achieve significant learning or discernible improvement.Further refinement are needed to improve the model's ability to learn and make accurate predictions for fractured bone detection.

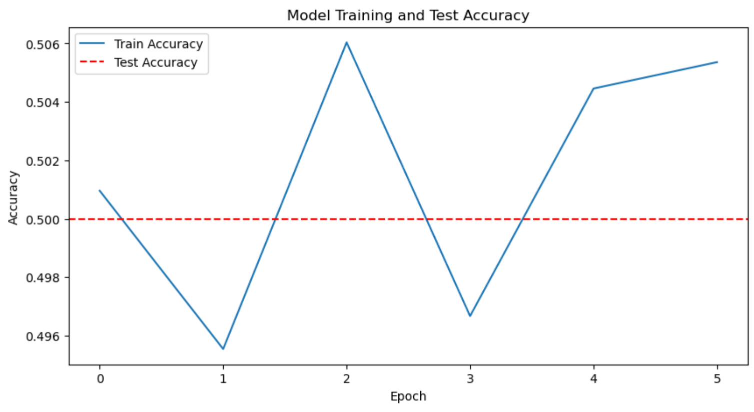

Figure 7: Accuracy of Transfer Learning

The evaluation process indicates :

- **Test Accuracy:** 0.5
- **Test Loss:** 0.6931

This suggests that the model's performance on the test dataset is sub-optimal, with an accuracy that is not much better than random guessing. Despite achieving a relatively low loss value, the accuracy remains poor, indicating that the model may be struggling to effectively classify the test data.

8 Vision Transformer

ViTs are adept at capturing long-range dependencies within images, which is particularly beneficial when analyzing medical images with complex structures like fractured bones. Traditional convolutional neural networks (CNNs) may struggle to effectively capture such dependencies across distant regions of an image, whereas ViTs can inherently model global context without relying on handcrafted hierarchical features. In fractured bone detection, ViTs have the potential to enhance both accuracy and efficiency, offering a promising avenue for further research and development in medical image analysis.

8.1 Architecture:

8.1.1 Adjusting Image Size for ViT

In this section, we resize the images to 224x224 pixels to align with the input requirements of the Vision Transformer (ViT) model. The ImageDataGenerators are re-initialized with the updated image size for both the training and testing datasets, ensuring compatibility with the ViT model.

8.1.2 Defining the Base Model

Here, we define the base model using EfficientNetB0, a convolutional neural network architecture pre-trained on the ImageNet dataset. By setting `include_top=False`, we exclude the fully connected layers at the top of the network, as custom layers will be added for our specific task. The input tensor is configured to accept images of size (224, 224, 3), matching the resized images from the data generators.

8.1.3 Adding Custom Layers

Custom layers are added on top of the base model to tailor it to the fractured bone detection task. This includes a Global Average Pooling layer to aggregate spatial information from the convolutional features, followed by a Dense layer with 512 neurons and ReLU activation for additional feature extraction. A Dropout layer with a dropout rate of 0.5 is incorporated to prevent overfitting. Finally, a Dense layer with a single neuron and sigmoid activation is appended for binary classification, producing the output predictions.

8.1.4 Constructing the Final Model

The final model is constructed by specifying the inputs (base model input) and outputs (predictions). It is then compiled with the Adam optimizer and binary cross-entropy loss, using accuracy as the evaluation metric.

8.2 Evaluation of Transformers:

The training process of the model unfolded over 10 epochs, where significant improvements in both accuracy and loss were observed. Initially, in the first epoch, the model achieved an accuracy of 80.13% with a loss of 0.4000. As training progressed, the accuracy steadily increased, reaching remarkable levels by the later epochs. By the fourth epoch, the model attained an impressive accuracy of 98.76% with a considerably reduced loss of 0.0351. However, in the subsequent epochs, slight fluctuations in accuracy were noticed, possibly indicating some degree of overfitting.

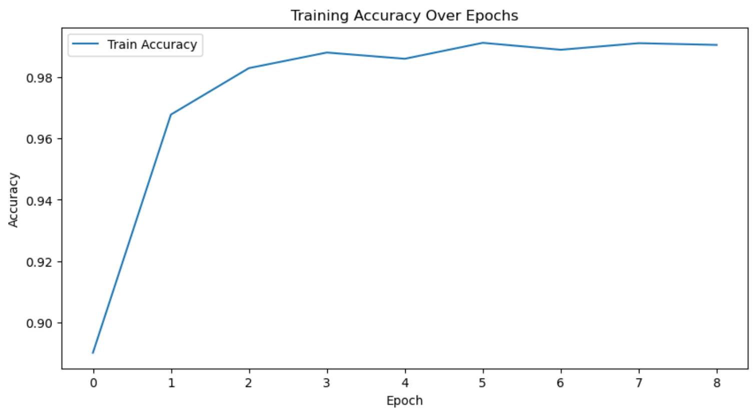

Figure 8: Accuracy of Transformers

The evaluation process indicates :

- **Test Accuracy:** 0.530
- **Test Loss:** 4.003

The test accuracy suggests that the model performs slightly better than random guessing, the significant disparity between training and test accuracy, along with the high test loss, indicates that the model may not generalize well to unseen data.

9 Conclusion

Our exploration of various models for fractured bone detection reveals distinct advantages and limitations across different architectures. The Multilayer Perceptron (MLP) demonstrated simplicity and ease of implementation but struggled to capture complex image features, resulting in suboptimal performance.
Convolutional Neural Networks (CNNs) exhibited superior performance by effectively extracting hierarchical features from images, showcasing their suitability for image classification tasks like bone fracture detection.
Transfer Learning, particularly with models like VGG16 and EfficientNetB0 pretrained on large datasets like ImageNet, facilitated the extraction of meaningful features, enhancing the model's ability to generalize to medical image datasets with limited samples. Vision Transformers introduced a novel approach leveraging self-attention mechanisms, showing promise in image classification tasks.
While each approach presents its merits, further research and experimentation are necessary to optimize model performance and address challenges such as overfitting, ensuring robust and reliable fractured bone detection systems in clinical settings.

10 Reference

[1] Dataset Link:

Bone Fracture Detection Dataset on Kaggle.