Final Report

Level 4

Pediatric Bone Age Detection and Adult Height Prediction Using Hand and Wrist Xray Images

Group Name: Nexero

194039R	Dushani R.P.U.
194057U	Hettiarachchi I.H.S.C
194181T	Wickramarachchi H.R.

Faculty of Information Technology

University of Moratuwa

June 2024

Final Report

Level 4

Pediatric Bone Age Detection and Adult Height Prediction Using Hand and Wrist Xray Images

Group Name: Nexero

194039R	Dushani R.P.U.
194057U	Hettiarachchi I.H.S.C
194181T	Wickramarachchi H.R.

Supervised By: Dr. Upeksha Ganegoda

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Honors Degree of Bachelor of Science in Information Technology.

August 2024

Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of Student(s)	**Signature of Student(s)**
Dushani R.P.U.	.....
Hettiarachchi I.H.S.C	.....
Wickramarachchi H.R.	.....

Date:.....31/08/2024.....

Supervised by:

Name of Supervisor	**Signature of Supervisor**
Dr. Upeksha Ganegoda	.....

Date:.....01/09/2024.....

Dedication

We dedicate our project, “Pediatric Bone Age Detection and Adult Height Prediction Using Hand and Wrist X-ray Images,” to the Dean of the Faculty of Information Technology, Mr. B.H. Sudantha and our project supervisor, Dr. Upeksha Ganegoda. We extend our gratitude to the academic and non-academic staff of the Faculty of Information Technology, whose guidance and encouragement have been invaluable.

We also acknowledge the support of our fellow students, lecturers, and the educational system that has shaped our learning. This project is our contribution to the fields of pediatric bone age assessment, deep learning, and image processing.

Acknowledgment

We would like to extend our heartfelt gratitude to the following people who have helped and guided us in numerous ways to make the project a success.

First and foremost, we extend our heartfelt thanks to our supervisor, Dr. Upeksha Ganegoda, for her invaluable assistance, constant guidance, and unwavering motivation, which were instrumental in the successful completion of this project. Her expertise and encouragement have been a driving force in achieving our objectives.

We are immensely grateful to Dr. Ashani Wickramarachchi (MBBS, MD, MRCP), Consultant in Endocrinology at District General Hospital, Sri Lanka, for proposing the project topic and providing crucial insights into the challenges of manual bone age assessment in hospitals. Her guidance and support were pivotal in shaping the direction of our research.

Our sincere thanks also go to Dr. Sameera Wijayawardhana, Lecturer at the Department of Anatomy, Faculty of Medicine, University of Kelaniya, and Dr. Jerard Fernando, Radiologist, for their expert advice and invaluable contributions to our work, which greatly enhanced the quality and relevance of our project.

We would also like to acknowledge all the academic staff at the Faculty of Information Technology, University of Moratuwa, for imparting their knowledge and providing guidance throughout our undergraduate years. Their teaching has laid the foundation for the skills and understanding that made this project possible.

We also appreciate the non-academic staff of the Faculty of Information Technology, University of Moratuwa, for their support and assistance during our time at the university.

Finally, we are deeply grateful to our families, friends, and batchmates for their continuous encouragement and support throughout this journey. Their belief in us has been a source of strength and motivation. This revision maintains the original sentiments while improving readability and flow.

Abstract

The Automated Pediatric Bone Age Detection System represents a groundbreaking initiative in pediatric healthcare, leveraging deep learning techniques for accurate and comprehensive assessments. The system, which focuses on hand and wrist X-ray images, combines advanced image preprocessing and bone age prediction algorithms, resulting in a comprehensive approach to pediatric growth evaluation. The system predicts carpal and phalangeal bone ages and then seamlessly merges these predictions using a strong regression model. The study extends its scope beyond bone age prediction to adult height prediction utilizing proven methods such as the Bayley-Pineau and TW2 approaches. When these forecasts are combined, final bone age and height of the child is generated, providing healthcare practitioners with extensive information about a child's skeletal development, probable growth abnormalities, and useful guidance. The study is built on comprehensive literature reviews, thorough dataset gathering, and continuous modeling research. This integrated approach ensures an advanced and effective pediatric bone age assessment solution, set to improve the precision and depth of growth-related diagnostics in pediatric healthcare.

Table of Contents

Introduction 1
1.1 Introduction 1
1.2 Background and motivation 1
1.3 Problem in brief 2
1.4 Aim and objectives 3
1.4.1 Aim 3
1.4.2 Objectives 3
1.5 Proposed solution 3
1.5.1 Module 1: Bone age assessment using carpal bones 4
1.5.2 Module 2: Bone age assessment using phalangeal bones 5
1.5.3 Module 3: Adult height prediction module using calculated bone ages 5
1.6 Summary 6
Literature review 7
2.1 Introduction 7
2.2 Pediatric bone age assessment 7
2.3 Manual Bone age assessment methods 9
2.4 Greulich-Pyle (GP atlas method) 9
2.5 Tanner-Whitehouse (TW) method 11
2.6 Automated bone age assessment approaches 12
2.6.1 Deep Learning Algorithms: 13
2.6.2 Transfer Learning: 14
2.6.3 Multi-Modal Analysis: 14
2.6.4 Real-time Processing: 14
2.7 Carpal bone analysis for bone age prediction 14
2.8 Phalangeal bone analysis for bone age prediction 19
2.9 Adult height prediction from bone age 19
2.9.1 Existing Methods for Adult Height Prediction 21
2.9.1.1 Bayley-Pinneau Method 21
2.9.1.2 Roche-Wainer-Thissen Method 22
2.9.1.3 Tanner-Whitehouse Method 23
2.9.1.4 Khamis-Roche method 24
2.9.2 Novel Machine Learning Approaches 25
2.9.2.1 BoneXpert System 26

2.9.2.2 Growth Curve Comparison (GCC) Method 26
2.9.2.3 Spampinato et al. method 27
2.9.3 Effectiveness of Automates Systems in Predicting Adult Height 27
2.10 Summary 28
Technology Adapted 29
3.1 Introduction 29
3.2 Technologies adopted for implementation 29
3.2.1 Programming languages 29
3.2.2 Development environments / tools 29
3.2.3 Libraries and frameworks: 30
3.2.4 Pre-trained Models: 30
3.2.5 Image Annotation Tools: 31
3.2.6 Machine Learning Techniques: 31
3.2.7 Version controlling: 31
3.3 Summary 31
Approach 32
4.1 Introduction 32
4.2 System Overview 32
4.3 Users 32
4.4 Inputs 33
4.5 Process 34
4.5.1 Carpal bone age assessment 34
4.5.1.1 Data Preprocessing 34
4.5.1.2 Hand Segmentation 35
4.5.1.3 Hand Alignment 35
4.5.1.4 Carpal Bone Region Segmentation 35
4.5.1.5 Carpal Bone Boundary Refinement 35
4.5.1.6 Model Training 35
4.5.2 Phalangeal bone age assessment 36
4.5.2.1 Data Loading and Preprocessing 36
4.5.2.2 Image Preprocessing 36
4.5.2.3 Dataset Preparation 37
4.5.2.4 Model Building and Training 37
4.5.2.5 Prediction and Visualization 37
4.5.3 Adult height prediction 38
4.5.3.1 Data Loading and Preprocessing 38
4.5.3.2 Model Building and Training 38

4.6 Outputs 39
4.7 Summary 41
Analysis and Design 42
5.1 Introduction 42
5.3 Carpal bone age assessment module 43
5.4 Phalangeal bone age assessment module 45
5.5 Height prediction module 47
5.6 Summary 49
Implementation 49
6.1 Introduction 49
6.2 Carpal bone age prediction module 49
6.2.1 Data preprocessing 49
6.2.2 Image preprocessing 54
6.2.3 Segmentation module 60
6.2.4 Hand region segmentation 63
6.2.5 Carpal bone segmentation module 73
6.2.6 Model Building and Training 77
6.2.7 Model Testing 81
6.3 Phalangeal bone age prediction module 82
6.3.1 Data Loading and Processing 82
6.3.2 Image Preprocessing and Dataset Preparation 84
6.3.3 Model Building and Training 87
6.3.5 Prediction 90
6.3.6 Attention Map Visualization 91
6.4 Adult height prediction model 94
6.4.1 Data Preprocessing 94
6.4.2 Adult height prediction using Bayley-Pineau Method 99
6.4.3 Analysis of the results from BP method 103
6.4.4 Enhancing prediction using Neural Network Model 105
6.5 Summary 109
Evaluation 110
7.1 Introduction 110
7.2 Carpal bone Age Prediction Module 110
7.3 Phalangeal Bone Age Prediction Module 113
7.4 Adult Hight Prediction Module 116
7.5 Summary 119
Discussion 120

8.1 Introduction 120
8.2 Carpal bone age prediction module 120
8.3 Phalangeal bone age prediction module 122
8.4 Adult height prediction module 123
References 126
Appendix A - Individual Contribution 130
194039R – Dushani R.P.U 130
194057U – Hettiarachchi I.H.S.C. 131
194181T Wickramarachchi H.R. 133

List of Figures

Figure 1: A left hand and wrist radiograph 8
Figure 2: Skeletal Development of a Hand 9
Figure 3: Estimate skeletal age using Greulich–Pyle method 10
Figure 4: TW3-related regions of interest. TW3, Tanner-Whitehouse 3 12
Figure 5: General architecture of deep learning methods for bone age assessment 13
Figure 6: Description of carpal bones in hand radiograph 15
Figure 7: Change of carpals with skeletal maturity from 0 years to 6 years. 16
Figure 8: High-Level Review of the System 42
Figure 9: High level architecture for carpal bone age assessment module 45
Figure 10: High level architecture for phalangeal bone age assessment module 47
Figure 11: High level architecture for adult height prediction module 48
Figure 12: Loading datasets to data frames 51
Figure 13: Count male and female data in the dataset 51
Figure 14: Filter dataset based on age and gender 51
Figure 15: Head of male and female data frames 52
Figure 16: Maximum and minimum ages in both datasets 52
Figure 17: Distribution of bone ages for each gender 53
Figure 18: Compare the bone age distributions between genders 54
Figure 19: Outputs of Frequency domain filtering 56
Figure 20: Histogram of original image before and after FDT 57
Figure 21: Frequency domain filtering code 58
Figure 22: Output results of CLAHE and biliteral filtering 60
Figure 23: Binarization code 61
Figure 24: Output results of binarization 61
Figure 25: Output results of generated hand masks 62
Figure 26: Code for hand mask generation 63
Figure 27: Output for segmented images 64
Figure 28: Code for segmenting hand regions 64
Figure 29: Code for finding contours 65
Figure 30: Code for drawing convex hull 65
Figure 31: Code for finding convexity defects 66
Figure 32: Output results of contour, convex hull and convexity defects 67
Figure 33: Code for finding palm center 68
Figure 34: Output images for palm center detected images 69
Figure 35: Output results of hand rotation steps 70
Figure 36: Code for drawing lines for rotation alignment 71
Figure 37: Code for rotating the image 71
Figure 38: Code for resizing the image 71
Figure 39: Output for annotated hand regions 73
Figure 40: Code for hand region annotation 73
Figure 41: Output for segmented lower hand regions 74
Figure 42: Outputs for carpal bone segmentation 75
Figure 43: Code for carpal bone segmentation 76
Figure 44: Code for image resizing 78
Figure 45: Code for data augmentation 78
Figure 46: Code for building CNN model 80
Figure 47: Code for model training 80
Figure 48: Model training (first 10 epoch) 80

Figure 49: Plot training & validation loss values 81
Figure 50: Code for model testing 82
Figure 51: Code for Importing Libraries 83
Figure 52: Code for Data Loading and Preprocessing 83
Figure 53: Code for Image Preprocessing and Dataset Preparation 86
Figure 54: Results of Image Preprocessing 87
Figure 55: Code for Dataset Preparing 88
Figure 56: Code for Building and Compiling Model 89
Figure 57: Code for training model 90
Figure 58: Code for Prediction 91
Figure 59: Code for Attention Map Generation 93
Figure 60: Results of Attention Map 93
Figure 61: Load datasets 94
Figure 62: Describe the dataset 95
Figure 63: Visualize the dataset 95
Figure 64: Basic information about the dataset 95
Figure 65: Check for missing values 96
Figure 66: Plot the histograms for each attribute and target values 96
Figure 67: Plot boxplots for each attribute 97
Figure 68: Pair plots 98
Figure 69: Heat map for attributes 99
Figure 70: BA delay calculation and divide data 100
Figure 71: Code to convert skeletal age to BP format 101
Figure 72: Code to get PMH and predict height 102
Figure 73: Code to apply BP tables 103
Figure 74: Calculate deviation 103
Figure 75: Summary statistics 103
Figure 76: Histogram of height prediction deviation 104
Figure 77: Scatter plot between predicted vs Actual adult Height 105
Figure 78: Code for data preparation 106
Figure 79: Code for training model for female dataset 107
Figure 80: Code for training model for male dataset 107
Figure 81: Code for male model 108
Figure 82: Code for female model 109
Figure 83: Carpal bone model evaluation results 110
Figure 84: Predicted vs. Actual Bone Age Scatter Plot 112
Figure 85: Distribution of Predicted vs. Actual Bone Ages 113
Figure 86: Validation loss and MAE 113
Figure 87: Prediction Error Distribution 114
Figure 88: Actual vs. Predicted Bone Age 114
Figure 89: Grad-CAM Visualization 115
Figure 90: Results of the Predicted values for Test Dataset 116
Figure 91: Codes for height prediction model evaluation and RMSE, MAPE calculations 117
Figure 93: values for male and female models 118
Figure 94: Predicted vs Actual Height Scatter Plots for male & female 118

List of Tables

Table 1: Overview of recent studies of carpal bone analysis 17
Table 2: Bone age methods & adult height prediction methods 20
Table 3: Comparison of different heigh prediction approaches 28
Table 4: Inputs for carpal bone age assessment module 33
Table 5: Inputs for phalangeal bone age assessment 33
Table 6: Inputs for adult height prediction 34
Table 7: Outputs for carpal bone age assessment 39
Table 8: Outputs for Phalangeal bone age assessment 40
Table 9: Outputs for Adult height prediction 41
Table 10: Bone age prediction for module 01 111

Chapter 1

Introduction

1.1 Introduction

Pediatric bone age assessment is a crucial aspect of clinical pediatrics, providing valuable insights into a child's growth and development [1] [2] [3] [4]. Traditional methods involve the manual examination of hand and wrist X-ray images by skilled radiologists, which can be both time-consuming and subjective [1] [2] [3] [4]. To address these challenges, this paper introduces the development of a hybrid automated Pediatric Bone Age Prediction System. This system utilizes image processing techniques and deep learning, particularly convolutional neural networks (CNNs), to analyze carpal and phalangeal bones for accurate bone age prediction [2] [3]. Additionally, it provides predictions of maturity height based on calculated bone ages, which is essential for early detection of growth disorders, guiding medical interventions, and monitoring treatment efficacy in pediatric and orthopedic [1] [2] [4].

1.2 Background and motivation

Traditional bone age assessment methods, such as the Greulich-Pyle (GP) and Tanner-Whitehouse (TW) methods, involve manual examination of hand and wrist X-ray images. The GP method compares X-ray images to a standard atlas, while the TW method requires a detailed analysis of multiple skeletal regions. Despite the simplicity of the GP method, it remains time-consuming and relies heavily on the radiologist's expertise, leading to potential inconsistencies and subjectivity in assessments. The TW method, though more comprehensive, is less frequently used due to its complexity [1] [5] [6].

These limitations highlight the need for an automated solution to enhance the accuracy and efficiency of bone age assessments. By leveraging image processing techniques and deep learning, specifically convolutional neural networks (CNNs), we can develop a system that provides consistent and objective evaluations. This automation will

reduce the workload on radiologists, minimize human error, and ensure standardized assessments [5] [6].

The motivation behind this project is to address the inefficiencies and potential inaccuracies of manual bone age assessment. By exploring the latest advancements and integrating them into our system, we aim to contribute to the ongoing evolution of medical image analysis, particularly in the context of pediatric bone age assessment. An automated system can expedite the diagnostic process, allowing for quicker and more reliable evaluations. Furthermore, the system's ability to predict maturity height based on bone age is crucial for early detection of growth disorders and effective medical intervention [1].

By considering above, this research project focuses on the development of an Automated Pediatric Bone Age Prediction System that utilizes advanced image processing and deep learning algorithms to analyze X-ray images of the hand and wrist to improve the quality of pediatric healthcare, which will ultimately benefiting both patients and healthcare providers [1] [5] [6] [7] [8].

1.3 Problem in brief

Assessing the bone age of children is essential for evaluating their growth and development, identifying growth abnormalities, and tracking the effectiveness of growth-modifying therapies. However, the manual bone age assessment methods, such as the Greulich-Pyle (GP) and Tanner-Whitehouse (TW) approaches, face several significant challenges.

The GP method, which is the most commonly used approach in clinical practice, relies on comparing X-ray images to standard atlases. This process is not only time-consuming but also prone to human error and subjectivity. Radiologists are often required to repeatedly consult reference materials, particularly in high-traffic healthcare settings, which places a considerable burden on medical professionals and introduces variability into the assessment process [9] [10].

The TW method is a more complex method that requires the identification and analysis of multiple skeletal regions, including the carpal bones, radius, ulna, and phalanges. It involves a scoring system that assigns numerical values to the maturity of each bone,

which makes it complex and time consuming. Due to the specialized training and expertise required, as well as the increased time and resource demands, physicians do not generally use it for bone age assessments [9] [10] [3].

Furthermore, current bone age assessment systems are limited in their ability to accurately predict adult height based on the calculated bone age. Traditionally, adult height prediction has relied on manual methods, such as the Bayley-Pinneau and Tanner-Whitehouse-Rutland (TWRH) methods, which are based on regression models and reference data. These manual approaches can be imprecise, as they do not account for individual variations and may not accurately capture the complex relationship between bone age and final adult height. This limitation can make it difficult for early detection of growth disorders and the effective planning of medical interventions [9] [6].

1.4 Aim and objectives

1.4.1 Aim

The primary aims of this research project are to develop an automated, deep learning-based system for accurate pediatric bone age prediction from hand and wrist X-ray images, utilizing a hybrid approach that combines the analysis of carpal bones and phalangeal bones, and to incorporate the capability to predict maturity height based on the calculated bone ages.

1.4.2 Objectives

1. To predict the bone age using carpal bones in pediatric patients under age 7.
2. To predict bone age using phalangeal bones in pediatric patients.
3. To predict adult height of pediatric patients using bone age predictions.

1.5 Proposed solution

This proposed system aims to develop a hybrid automated pediatric bone age prediction system using hand and wrist X-ray images for pediatric patients. It incorporates three key modules which are,

- Bone age assessment using carpal bones
- Bone age assessment using phalangeal bones
- Adult height prediction module using calculated bone ages

Bone age assessments modules use hand and wrist Xray images as inputs and assess the bones ages using different image processing and deep learning techniques. These two modules use different bone types to assess bone ages which are carpal and phalangeal bones separately. Since in pediatric patients, carpal and phalangeal bones have different growth patterns, they can be used separately for more accurate bone age assessment [11] [12].

By analyzing the maturity of both carpal and phalangeal bones independently, the automated bone age prediction system can provide a more comprehensive evaluation compared to using only one set of bones or considering them together. This approach leverages the unique growth characteristics of these skeletal regions to enhance the overall accuracy and reliability of the bone age assessment. The height prediction module takes bone ages and gender as inputs and predicts the height of the pediatric patient in future.

1.5.1 Module 1: Bone age assessment using carpal bones

The aim of this module is to predict the bone ages of children under the age of 7 using the analysis of carpal bones. This module employs a series of image processing techniques, including filtering, smoothing, edge detection, and segmentation, to accurately segment the carpal bones from the hand and wrist X-ray images.

An important consideration in this task is that carpal bones begin to overlap around the ages of 7 in males and 5 in females. To address this, the module first extracts data relevant to these age groups separately for each gender. Subsequently, a unified dataset is created by integrating both gender-specific data. This combined dataset undergoes preprocessing, and the images are also prepared for segmentation of the carpal bones.

Following the segmentation of the carpal bones, the module utilizes deep learning techniques, specifically Convolutional Neural Networks (CNNs), to develop a bone age prediction model. A CNN-based model is trained on gender-specific labeled X-ray image datasets to learn the complex patterns and relationships between the carpal bone characteristics and the corresponding bone ages.

When a user provides an X-ray image and specifies the gender, the module will predict the bone age of the child. This approach ensures that the predictions are tailored to the

specific developmental patterns of each gender, leading to more accurate and reliable bone age assessments.

1.5.2 Module 2: Bone age assessment using phalangeal bones

In this module X-ray images are preprocessed by converting them to grayscale, applying gamma correction for brightness and contrast adjustment, CLAHE for enhanced contrast, image fusion, and noise reduction.

Phalangeal regions of interest (PROI) are extracted using a pre-trained Roboflow model, filtered to include relevant detections, and annotated. We utilize VGG16 as the base model, augmenting it with custom layers for fine-tuning, and freezing the base model layers to preserve pre-trained weights.

The model is compiled and trained, with its performance evaluated using loss and mean absolute error metrics. For prediction, we visualize attention maps using Grad-CAM, overlaying heatmaps on the original images to highlight significant regions influencing bone age predictions. This comprehensive approach leverages advanced image processing and deep learning techniques to ensure accurate and interpretable bone age predictions.

1.5.3 Module 3: Adult height prediction module using calculated bone ages

The Adult Height Prediction (AHP) system aims to accurately forecast the final adult height of children based on their bone age, current age, and current height, particularly adapted for the Asian demographic.

The proposed solution includes several critical steps: first, preprocessing and cleaning the dataset to address missing values and ensure standardized units; second, calculating the bone age delay for each entry to categorize children into normal, advanced, and delayed maturity groups; third, using distinct BP tables for males and females to predict adult height based on these categories; and finally, applying machine learning techniques to improve accuracy by training regression models on features like BA delay, BA category, and demographic-specific data.

The model's performance is validated using metrics like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) to ensure accuracy. By integrating traditional BP

tables with modern machine learning techniques, this solution aims to enhance adult height prediction accuracy and adapt it to diverse demographic groups.

1.6 Summary

This chapter has discussed an introduction to the research area, background and motivation, aim and objectives of the research. Furthermore, it discusses the proposed solution and its components. The rest of the report is as follows. Chapter 2 review existing works of bone age prediction of pediatric patients using hand and wrist Xray images. Chapter 3 describes the technologies adopted during the implementation of the system. Chapter 4 illustrates the approach to developing the system. Chapter 5 depicts the analysis and design of the proposed system and its sub modules. Chapter 6 discusses the implementation. Next chapter, chapter 7, gives the conclusion of the report and the future works of the system. Finally, the report will conclude with a list of references that was used during the implementation of the system

Chapter 2

Literature review

2.1 Introduction

This chapter illustrates the previous approaches that were used in similar research to problem and proposed solution mentioned in previous chapter. The system consists of three main modules and this chapter contains related work for each module of the system and system as a whole.

2.2 Pediatric bone age assessment

Bone age is an indicator of the skeletal and biological maturity of an individual, distinct from their chronological age calculated from date of birth. Assessing a child's bone age provides valuable insights into their growth and development and is an important tool in pediatric healthcare [13] [14].

Bone age assessment is widely used to diagnose and manage various pediatric conditions, including growth disorders, endocrine dysfunctions, and developmental delays. It allows clinicians to compare a child's skeletal maturity to population norms, identify abnormalities, and guide appropriate interventions. Tracking a child's bone age over time can also help predict their final adult height and monitor the effectiveness of treatments [13].

Hand and wrist x-rays are regularly utilized to assess bone age in children because the bones in the hand and wrist mature at a predictable rate, making them a good indicator of a child's biological age. The assessment is performed with a radiograph of the non-dominant hand, and the appearance of the carpal bones, metacarpal, phalanges, radius, and ulna are compared to standard atlases in the Greulich Pyle and Tanner Whitehouse manual approach *[14] [15]*.

As shown in Figure 1, six phalangeal regions, including the distal, middle, and proximal phalanges, as well as the seven carpal bones, can be seen. These regions are crucial in the assessment of bone age, providing key indicators of skeletal maturity.

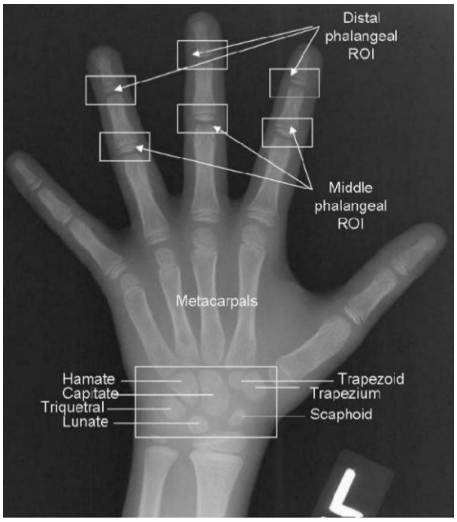


Figure 1: A left hand and wrist radiograph

The use of no dominant hand for bone age detection is due to the fact that it is less likely to be injured or impacted by external variables such as muscle development or injury. As the bones in the hand and wrist fuse and stop growing beyond the age of 18, they are not suitable for bone assessment in adults. Instead, alternative bones such as the clavicle will be used to determine adult bone age [15].

As illustrated in Figure 2, the development and growth of a child's carpal (wrist) and phalangeal (finger) bones can be observed over time. At birth, these bones are primarily composed of cartilage and are not yet fully formed or visible on an X-ray image. However, as the child grows and develops, the carpal bones begin to ossify (turn into bone) during the first two years of life, with small ossification centers becoming visible. The phalangeal bones also start to ossify, gradually increasing in size, density, and complexity throughout childhood and adolescence. By around 2-4 years old, the ossification centers become more distinct, and the bones take on their characteristic shapes. During childhood (4-12 years), the carpal bones become more intricate while the phalangeal bones elongate, with the ossification centers fusing and the bones becoming more dense and solid. Finally, by adolescence (12-18 years), the carpal and phalangeal bones reach their final adult size and shape, with the growth plates closing to indicate the end of bone growth. The figure demonstrates the progression of this

fascinating process, which is essential for understanding the growth and development of the human hand from birth to adulthood.

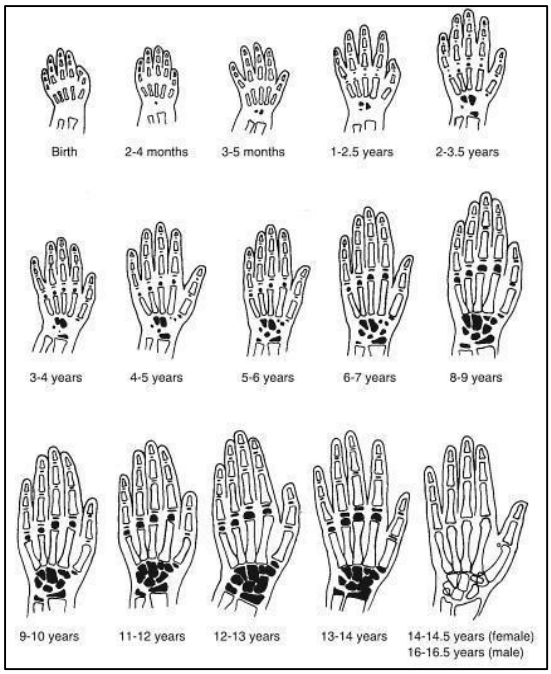


Figure 2: Skeletal Development of a Hand

2.3 Manual Bone age assessment methods

Pediatric bone age evaluation is critical in clinical pediatrics as it provides important insights into a child's biological maturity beyond chronological age. This bone age assessment, which is commonly performed using hand and wrist radiography, contributes to detecting growth problems and planning treatments. The most widely used method for assessing bone age is manual assessment of hand and wrist radiographs using standardized atlases. The two most common atlases used in the medical field are the Greulich-Pyle (GP) atlas and the Tanner-Whitehouse (TW) method.

2.4 Greulich-Pyle (GP atlas method)

The GP atlas was developed in 1959 and contains radiographic standards of the left hand and wrist for each age from birth to 19 years for girls and 20 years for boys. The radiographs in the atlas represent the average development for that age and sex. To use this method, the child's radiograph is visually compared to the standard radiographs in the atlas, and the bone age is assigned based on the closest matching image from the atlas [1].

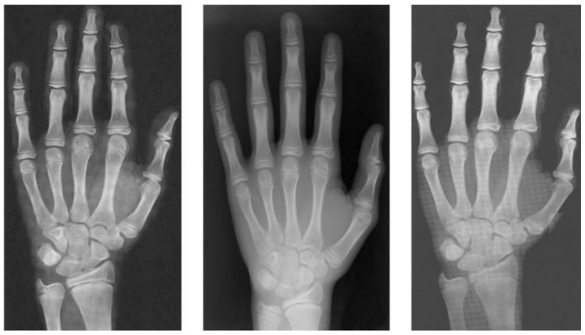


Figure 3: Estimate skeletal age using Greulich–Pyle method

Figure 3 illustrates the approach to estimating skeletal age using the Greulich–Pyle method. The left image features the Greulich–Pyle atlas standard 25, which corresponds to a skeletal age of 14 years for males. The middle image displays a hand radiograph of the investigated subject, which is assessed against the atlas standards and also indicates a chronological age of 14 years. On the right, the Greulich–Pyle atlas standard 26 is shown, representing a skeletal age of 15 years for males. The middle X-ray of the investigated subject definitely meets the criteria of Greulich–Pyle standard 25 (skeletal age: 14 years):

- The epiphysis of the radius and the epiphysis of all phalanges of the second, third, fourth, and fifth fingers have begun to cap their shafts.
- The sides of the epiphysis of the second to the fifth metacarpals are now aligned closely with the sides of their shafts. The growth cartilage plates are uniformly narrow, and some portions of the epiphysial shaft spaces are fuzzy.

However, the middle X-ray does not yet meet the criteria of Greulich–Pyle male standard 26 (skeletal age: 15 years):

- The epiphysis of the ulna is not as wide as its shaft and does not follow its contour closely.
- Fusion is not under way in the epiphysis of all distal phalanges.

Therefore, the skeletal age of the investigated subject is estimated to be 14 years, as evidenced by the clear alignment with Greulich–Pyle standard 25.

Advantages of this method are its simplicity and the wide acceptance. This method is straightforward and easy to understand. It is widely accepted and used globally in clinical practice [1] [4].

The main limitations of this method include subjectivity, time consumption and limited nature of the atlas. The accuracy of the assessment can vary based on the radiologist's experience and interpretation, leading to potential inconsistencies. Manually comparing X-rays with the atlas can also be time-consuming, especially in busy clinical settings. The GP atlas may not adequately represent the diverse population, as it was developed based on a specific demographic [1] [4].The GP atlas may not adequately represent the diverse population, as it was developed based on a specific demographic [1] [4] .

2.5 Tanner-Whitehouse (TW) method

The TW method, developed in 1962 by James Tanner and colleagues, involves scoring the maturity of individual bones in the hand and wrist rather than comparing the entire hand/wrist to standardized radiographs. In this method numerical scores called TW scores are assigned to each 20 bones of the hand and wrist based on its stage of maturation. The 20 bone scores are then combined into an overall skeletal maturity score, which is converted to a bone age [13] [12].

There are three main versions of the TW method:

- TW1 - Evaluates all 20 bones in the hand and wrist
- TW2 - Evaluates 13 "radius-ulna-short" (RUS) bones and 7 carpal bones separately
- TW3 - Evaluates only the 13 RUS bones

Figure 4 illustrates the regions of interest related to the Tanner-Whitehouse 3 (TW3) method, which is utilized for assessing skeletal age in children. This figure highlights the 20 bones that are critical in the TW3 method, which includes both the radius and ulna as well as the short bones of the hand.

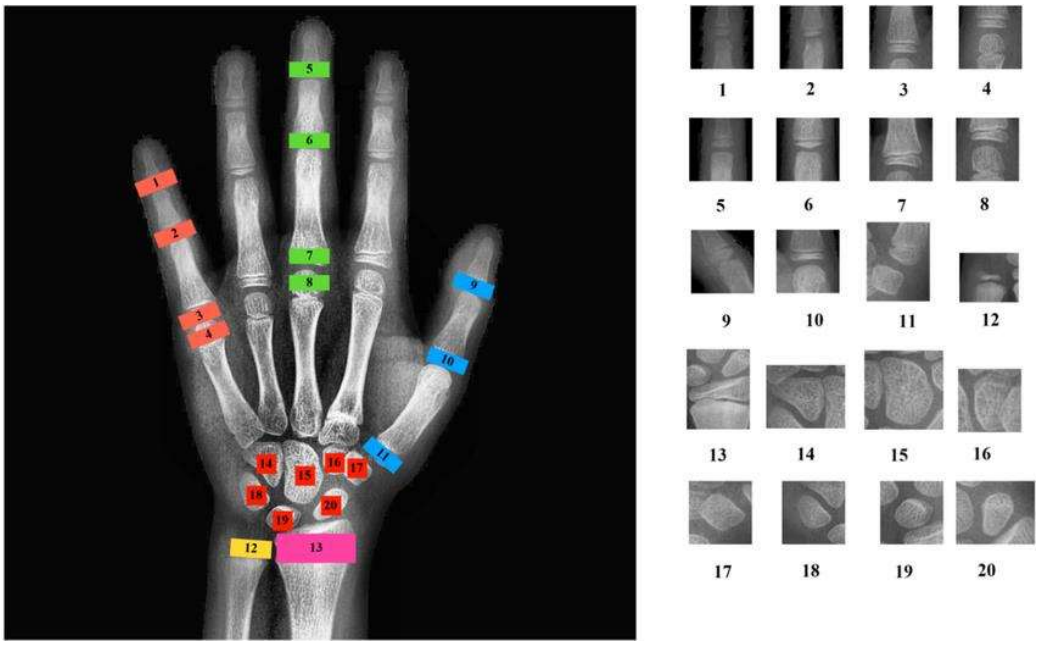


Figure 4: TW3-related regions of interest. TW3, Tanner-Whitehouse 3

This method is more objective and quantitative than the GP atlas method and provides separate scores for different bone groups, allowing assessment of regional maturity. It also allows for prediction of final adult height [12].

However, TW method is more complex and time-consuming to perform compared to the GP atlas. Reference data was obtained from European children in the 1950s-1960s, which may not be representative of modern, diverse populations and also secular trends in growth and development may have changed since the original data was collected [12] [13].

2.6 Automated bone age assessment approaches

The early attempts at automating bone age assessment were primarily focused on using computer vision and image processing techniques to analyze radiographs. These early systems were based on simple thresholding and edge detection algorithms, which were limited in their ability to accurately assess bone age due to the complexity of the images and the variability in bone morphology [12].

One of the earliest attempts at automating bone age assessment was the use of thresholding and edge detection algorithms to identify the epiphyseal plates in radiographs. These methods were able to segment the bones and identify the epiphyseal plates, but they were not accurate enough to provide reliable bone age assessments

[1,2]. Another approach involved using neural networks to classify the maturity of individual bones. While these methods showed some promise, they were still limited by the quality of the images and the variability in bone morphology.

Recent advancements in image processing techniques have significantly improved the accuracy and reliability of automated bone age assessment methods. These advancements include the use of deep learning algorithms, such as convolutional neural networks (CNNs), to analyze radiographs and predict bone age [8].

2.6.1 Deep Learning Algorithms:

- **Convolutional Neural Networks (CNNs)**: CNNs have been particularly effective in analyzing medical images, including radiographs [5] [8]. These networks can learn to identify subtle features and patterns in the images, which is crucial for accurate bone age assessment [5]. CNNs have been trained on large datasets of radiographs and corresponding bone age assessments, allowing them to learn to predict bone age with high accuracy [5] [8] .

  Figure 5 illustrates the general architecture of a deep learning method for bone age assessment. The process begins with an input X-ray image, which is fed into a convolutional network designed to extract relevant features from the image. These features are then processed through a regression network that interprets the extracted data to estimate the bone age. This systematic approach leverages the capabilities of deep learning to automate and enhance the accuracy of bone age assessments, providing a more efficient alternative to traditional methods.

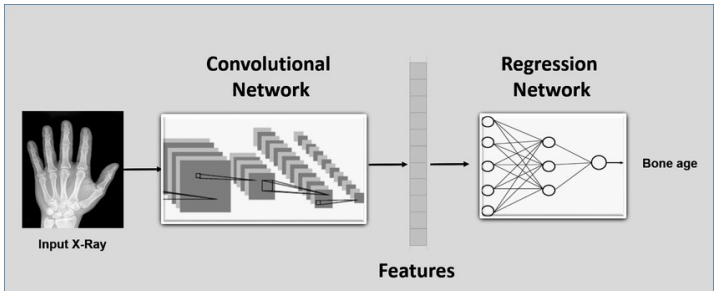


Figure 5: General architecture of deep learning methods for bone age assessment

2.6.2 Transfer Learning:

- **Pre-trained Models**: Pre-trained models, such as those developed for natural image classification tasks, can be fine-tuned for medical imaging tasks, including bone age assessment [8] [5]. This approach leverages the knowledge gained from large datasets and adapts it to the specific task at hand.

2.6.3 Multi-Modal Analysis:

- **Combining Multiple Imaging Modalities**: Recent research has explored the use of multiple imaging modalities, such as radiographs and MRI scans, to improve the accuracy of bone age assessment. This multi-modal approach can provide a more comprehensive view of skeletal maturity, leading to more accurate predictions [16].

2.6.4 Real-time Processing:

- **Efficient Inference**: The development of efficient inference algorithms has enabled real-time processing of medical images, which is crucial for clinical applications. This allows clinicians to receive rapid feedback on bone age assessments, facilitating timely interventions [5] [8].

2.7 Carpal bone analysis for bone age prediction

The carpal bones play a crucial role in assessing bone age, particularly in young children under the age of 7 [17] [18]. During this critical period of skeletal development, the carpal bones undergo significant changes and ossification, providing valuable insights into a child's overall skeletal maturity [18] .

Figure 6 showcases the carpal bones as seen in hand radiographs, highlighting all seven carpal bones. These bones include:

1. **Scaphoid**
2. **Lunate**
3. **Triquetrum**
4. **Pisiform**
5. **Trapezium**
6. **Trapezoid**
7. **Capitate**
8. **Hamate**

This figure provides a clear visual representation of the carpal bones, which are essential for the structure and function of the wrist, connecting the forearm to the hand and facilitating a wide range of movements.

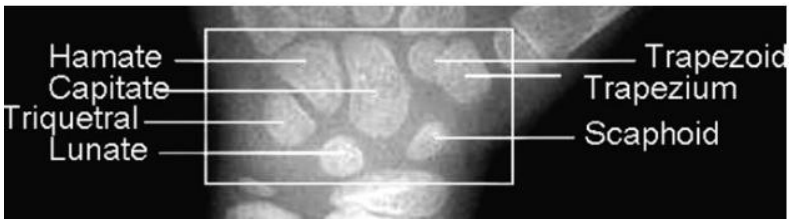


Figure 6: Description of carpal bones in hand radiograph

In infants and toddlers, the carpal bones are often the first to undergo ossification and exhibit visible changes on radiographic images [18]. Analyzing the appearance and development of these small bones can help clinicians accurately determine bone age in young children, where the use of traditional methods like the Greulich-Pyle (GP) atlas may be limited [17] .

Furthermore, the carpal bones can serve as an important indicator of skeletal maturity in cases where the epiphyseal plates of the long bones have not yet fully developed, as is common in younger children [18]. By focusing on the carpal bone analysis, clinicians can obtain a more comprehensive assessment of a child's skeletal age, leading to earlier detection of growth abnormalities and timely interventions.

Figure 7 illustrates the changes in carpal bones with skeletal maturity from 0 to 6 years of age. At **0 years**, no carpal bones are visible as they are still cartilaginous. By **1 year**, the capitate and hamate bones begin to ossify, marking the first appearance of carpal bones. At **2 years**, the triquetrum starts to appear, indicating further skeletal development. By **3 years**, the lunate bone becomes visible, adding to the growing structure of the wrist. At **4 years**, the scaphoid bone begins to ossify, followed by the trapezium and trapezoid, which appear around **5 years**. Finally, by **6 years**, the carpal bones are more defined, with the trapezium and trapezoid clearly visible, while the pisiform remains the last carpal bone to ossify, typically appearing around the age of 12. This progression highlights the critical stages of carpal bone development during early childhood.

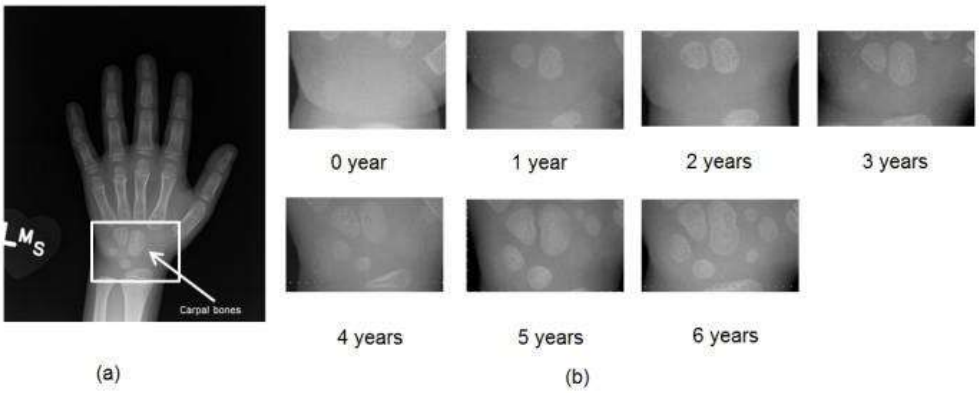


Figure 7: Change of carpals with skeletal maturity from 0 years to 6 years.

Several studies have explored the use of carpal bone analysis for improved bone age assessment, particularly in young pediatric populations [17] . Research has developed quantitative carpal bone analysis methods to quantify the size, shape and maturation of individual carpal bones using image processing techniques [17]. These measurements can then be used to predict the overall bone age. One such approach involves scoring the maturity of each carpal bone based on its appearance and combining the scores to determine the skeletal age.

With the advancements in computer vision and deep learning, researchers have explored automated methods to analyze carpal bones for bone age assessment. These techniques leverage convolutional neural networks (CNNs) and other machine learning algorithms to accurately identify and evaluate the carpal bones, reducing the subjectivity and variability inherent in manual assessments [18]. Some studies have investigated the integration of carpal bone analysis with other imaging modalities, such as radiographs and MRI scans, to provide a more comprehensive assessment of skeletal maturity. By combining information from multiple sources, these multimodal approaches have demonstrated improved accuracy in predicting bone age, especially in younger children [17].

The incorporation of carpal bone analysis, whether manual or automated, has been shown to enhance the accuracy and reliability of bone age assessment, particularly in the critical early years of childhood development [18]. This approach is particularly valuable in cases where traditional methods may be less effective, making it a valuable tool in pediatric healthcare.

Table 1 provides an overview of recent studies focused on carpal bone analysis.

Table 1: Overview of recent studies of carpal bone analysis

Study	Preprocessing technique	Carpal bone analysis method	Key findings	Limitations
F. Canovas et al [19]	3D CT imaging	Morphological and positional analysis of carpal bones in 3D	Detailed 3D characterization of normal carpal bone anatomy	Limited to a small sample size of 18 adults
Pan Lin et al [20]	Anisotropic diffusion filtering	Carpal bone segmentation using region-based level set method	Accurate and robust carpal bone segmentation by incorporating region statistical information	Complexity of skeletal structure and variability between subjects makes fully automatic segmentation challenging
Aifeng Zhang et al [21]	Adaptive thresholding, Anisotropic diffusion filtering, Canny edge detection	Canny edge detection followed by knowledge-based morphological operations for segmentation and feature extraction.	Reliable feature extraction for carpal bones in children under 7 significantly improves bone age assessment accuracy.	Less effective when carpal bones overlap,
Ewa Pietka et al [22]	Dynamic thresholding, Mathematical Morphology	Identify and describe objects in CROI using parameter such as area, perimeter, center of gravity, sector-based object classification.	Successfully identify and describe carpal bones using multiple features	Limited to children under 9 (boys) and 8 (girls); Affected by noise in low-contrast images.
Pengyi Hao et al [23]	Median filter Canny edge detection	ROI segmentation, smoothing, edge detection, and object refinement using eccentricity	High accuracy in bone age assessment; 2.75 months average discrepancy; 90.15% accuracy within 6 months for	Designed for young children (0-6 years); Small dataset (432 radiographs);

	Morphological operations	and solidity criteria to isolate carpal bones.	males; 99.43% accuracy within 1 year; Reduced assessment time and inter-observer variability.	Accuracy affected by image quality; Not tested on different populations; Heuristic thresholds may need adjustments.
Chien-Chuan Ko et al [24]	Contrast enhancement using histogram equalization and Sobel operator-based edge detection.	Region-based segmentation with moment-preserving thresholding, morphological processing, and region-growing labeling.	Accurate and reliable geometric parameter estimation of carpal bones for skeletal growth assessment.	Ineffective for patients older than 9-12 years due to bone overlap and requires anatomical knowledge for certain bones.
Liyilei Su et al [25]	CROI extraction using grayscale thresholding and geometric features.	Integrated region-based (adaptive thresholding) and boundary-based (Canny edge detection) segmentation with optional refinement.	Accurate carpal bone segmentation with high average Dice coefficient (0.976) using watershed refinement.	Relies on accurate CROI extraction, sensitive to image quality, limited to non-overlapping bones, requires further validation.

2.8 Phalangeal bone analysis for bone age prediction

Evaluation of skeletal maturity through hand and wrist X-rays is a common clinical procedure to assess a child's biological age [13]. The appearance and development of the phalanges, metacarpals, and carpal bones are key indicators of skeletal maturity that can be used to determine bone age [13] [4].

Traditional methods of bone age assessment, such as the Greulich-Pyle and Tanner-Whitehouse atlases, rely on visual comparison of X-ray images to standardized reference images [13]. However, these manual methods are time-consuming and prone to inter- and intra-observer variability [4].

In recent years, there has been growing interest in automated, computer-assisted methods of bone age assessment that analyze phalangeal bones [4]. These techniques use machine learning algorithms, particularly convolutional neural networks, to automatically detect and analyze the shape and size of the phalanges in hand X-ray images [4]. Studies have shown that these automated phalangeal analysis methods can achieve high accuracy in predicting bone age, with mean absolute differences from expert radiologists of around 4-5 months [4].

Beyond just predicting overall bone age, some automated systems also analyze the carpal bones and provide separate bone age assessments for the phalanges and carpals [26]. This can be useful, as the development of the phalanges and carpals do not always perfectly align and analyzing them separately may provide additional diagnostic information [26].

Overall, the literature indicates that automated phalangeal bone analysis is a promising approach for accurate, objective, and efficient bone age assessment in children [13] [4] [26]. As these techniques continue to be refined and validated, they may become the new standard for clinical bone age evaluation, replacing manual atlas-based methods.

2.9 Adult height prediction from bone age

Adult height prediction is a crucial aspect in clinical practice and research, particularly in the diagnosis and management of growth disorders. [27]Accurate prediction of adult height is essential for planning and monitoring treatment strategies, as well as for providing realistic expectations to patients and their families [27].

Several methods have been developed over the years to predict adult height, including the Bayley-Pinneau method, Roche-Wainer-Thissen method, and Tanner-Whitehouse method. [27]A new generation of methods for bone age assessment has appeared approximately every twenty years, and each new bone age assessment method has led to a new generation of adult height prediction methods.

Table 2: Bone age methods & adult height prediction methods

Bone age Generations		**Bone age methods**	**Adult height prediction methods**	**Bone age used in adult height prediction**
1	1946 – 56	Todd/ Greulich - Pyle	Bayley-Pinneau (1946/52)	Todd/GP
			RWT (1975)	GP bone-specific
2	1962 – 83	Tanner Whitehouse, TW1, TW2, TW3	TW Mark I (1975)	TW2
			TW Mark II (1983)	TW2
			TW3 (2001)	SMS
3	1987 – 93	Fels	RWT/Khamis (1993)	Fels
4	2008 -	BoneXpert	BoneXpert (2009)	BoneXpert

These conventional methods mentioned in Table 2 were based on data from normally growing children and have been widely used in clinical practice. [27]However, they often have systematic errors when applied to children with growth disorders or short stature. [27] [28]

Despite their widespread use, these methods have several limitations. They typically require a large number of diverse measurements, which can be time-consuming and cumbersome [27].Additionally, the calculations involved are often complex, making them difficult to apply in routine clinical settings. Furthermore, these methods are often ethnically specific and not generalizable, which can lead to inaccurate predictions for children from diverse backgrounds [27].

In recent years, researchers have explored the use of machine learning techniques to improve adult height prediction. [27]One notable example is the Growth Curve Comparison (GCC) method, which was developed using a large cohort of over 16,000 Slovenian schoolchildren. [27] This method uses machine learning algorithms to compare an individual's growth curve to a reference growth curve, allowing for more

accurate predictions. The GCC method has been shown to have lower prediction error compared to other non-intrusive methods, making it a promising approach for clinical use. [27]

Despite these advancements, there is still a need for more accurate, non-intrusive, and easily applicable techniques for predicting adult height, particularly for children with growth disorders. [27] [28] Future research should focus on developing methods that are more robust, efficient, and generalizable, allowing for better clinical decision-making and patient care. [27] [28]

2.9.1 Existing Methods for Adult Height Prediction

2.9.1.1 Bayley-Pinneau Method

The Bayley-Pinneau (BP) method is one of the most widely used techniques for predicting adult height in children. This method was developed in the 1950s by researchers Bayley and Pinneau using data from normally growing children. The core premise of the Bayley-Pinneau method is that adult height can be estimated based on the child's chronological age, current height, and bone age.

The BP method was first formulated by Bayley in 1946 and was based on Todd’s method for the determination of bone age. The Greulich-Pyle (GP) bone age method is a refinement of Todd’s atlas method and is based on data from the Brush foundation study of children from Ohio. The GP atlas contains plates corresponding to a series of selected ages, each plate representing a child with the median maturity at that age [29].

Bayley and Pinneau revised their method in 1952 using the Greulich-Pyle atlas, thus completing the famous Bayley-Pinneau (BP) method. This method was validated on 192 Berkeley children who were followed longitudinally and on another 46 children from Berkeley [29]. The second and final edition of the GP atlas appeared in 1959, and it did not require an update of the BP method, as the 1952 article on the BP method appears as an appendix to the 1959 edition of the GP atlas [29].

The BP method estimates adult height based on the child's chronological age, current height, and bone age. Bone age is assessed from an X-ray of the child's left hand and wrist using the Greulich-Pyle standards. The child's current height and this assessed bone age are then used to look up the predicted adult height in reference tables provided by Bayley and Pinneau [30].

While the BP method has been extensively used in clinical practice, it has several important limitations that have been highlighted in the literature. First, the requirement for bone age assessment via hand-wrist X-rays is a significant drawback, as it is time-consuming and requires specialized expertise. This makes the BP method less practical for routine use in many clinical settings [30].

Additionally, the BP tables were developed using data from children in the 1930s and 1940s. Given the secular trends in growth observed over the past several decades, the applicability of these outdated reference data to modern populations has been called into question [30]. Studies have found that the BP method may systematically over- or underestimate adult height, particularly when applied to children with growth disorders or short stature [30] [31].

For example, one study found that the BP method overestimated final adult height by 5.3 cm in boys with idiopathic short stature [30] [31]. This lack of accuracy is a major limitation, as precise height prediction is crucial for appropriate clinical management and treatment planning for children with growth-related conditions. [30]

Furthermore, the BP method was developed using data from specific populations and may not generalize well to children from diverse ethnic backgrounds. This ethnic specificity can lead to inaccuracies when applying the BP technique to individuals outside the original reference populations [30].

In summary, while the Bayley-Pinneau method has been a widely used tool for adult height prediction, it faces several important limitations. These include the requirement for bone age assessment, the use of outdated reference data, systematic errors when applied to children with growth disorders, and a lack of generalizability across diverse populations. The emergence of more sophisticated prediction methods, such as the GCC approach, highlights the need for continued research and development of accurate, efficient, and clinically applicable techniques for forecasting adult height in children [30] [31].

2.9.1.2 Roche-Wainer-Thissen Method

This method was developed by Roche, Wainer, and Thissen to estimate adult stature based on data collected at a single childhood examination. The RWT method is known for its simplicity and ease of use, making it a popular choice in clinical practice. The RWT method was designed to predict adult height from a single set of measurements,

including chronological age, current height, weight, and mean height of the parents. This approach aims to leverage the relationship between these variables to forecast the individual's final adult stature [27].

To use the RWT method, clinicians collect chronological age, current height, weight and mean parental height. These measurements are then used to look up the predicted adult height in reference tables provided by Roche, Wainer, and Thissen. The RWT method is based on the assumption that the child's growth trajectory can be predicted from these initial measurements to predict their adult height [32].

While the RWT method is simple and widely used, it has several limitations. The method relies on the availability of parental height data, which can be challenging to obtain in many clinical settings. This limitation reduces the applicability of the RWT method in practice. The RWT method was developed using data from specific populations and may not generalize well to children from diverse ethnic backgrounds. This ethnic specificity can lead to inaccuracies when applying the method to individuals outside the original reference populations [33].

Studies have shown that the RWT method can have systematic errors when applied to children with growth disorders or short stature. For example, one study found that the RWT method overestimated final adult height by 1.33 cm in boys with idiopathic short stature. Also this method was developed using data from children in the 1930s and 1940s. Given the secular trends in growth observed over the past several decades, the applicability of these outdated reference data to modern population can be unsuitable.

2.9.1.3 Tanner-Whitehouse Method

The Tanner-Whitehouse (TW) method was developed by Tanner and Whitehouse in the 1970s and has undergone several revisions since then, with the most recent version being TW3 in 2001 [34]. The TW method estimates adult height based on the child's chronological age, current height, and bone age. Bone age is assessed from hand and wrist X-ray using the Tanner-Whitehouse standards [34] [35]. The child's current height and bone age are then used to look up the predicted adult height in reference tables provided by Tanner and Whitehouse [34].

The TW method was developed using data from the Harpenden Growth Study, which followed children from birth to adulthood [34]. The method was designed to account

for the variability in growth patterns by providing separate predictions for early, average, and late maturer [36].

Over the years, the TW method has undergone several revisions to improve its accuracy and applicability.

1. TW Mark I (1975) - The original version of the TW method, which used a scoring system to assess bone age.
2. TW Mark II (1983) - A revised version that simplified the bone age assessment and provided updated reference tables.
3. TW3 (2001) - The most recent version, which incorporates data from more diverse populations and provides improved accuracy for children with growth disorders or short stature.

While the TW method has several limitations such as determining bone age from hand-wrist X-rays which can be time-consuming and requires specialized expertise, making the TW method less practical for routine clinical use. The TW method was developed using data from specific populations and may not generalize well to children from diverse ethnic backgrounds [34]. Studies have shown that the TW method can have systematic errors when applied to children with growth disorders or short stature [34].

2.9.1.4 Khamis-Roche method

The Khamis-Roche (KR) method is a non-intrusive approach for predicting adult height in children that was developed as a modification of the Roche-Wainer-Thissen (RWT) method. The key innovation of the KR method is that it does not require the assessment of skeletal age, which was a limitation of the original RWT technique.

The KR method was developed by Khamis and Roche in the 1990s using longitudinal data from the Fels Longitudinal Study [37]. The goal was to obtain reliable and accurate predictions of adult stature in white American children without the need for determining skeletal age.

The KR method estimates adult height based on the child's chronological age, current height, weight, and mean height of the parents. By eliminating the requirement for skeletal age assessment, the KR method becomes a more practical and less intrusive approach compared to other conventional adult height prediction techniques [38].

Studies have shown that the prediction errors of the KR method are only slightly larger than those of the RWT method, which does use skeletal age as a predictor variable. This suggests that the KR method can provide reasonably accurate adult height predictions without the need for the more complex and time-consuming bone age evaluation. One study found that the average error of the KR method was 1.33 cm larger than the RWT method when applied to a sample of white American children without pathological conditions. This slight deterioration in accuracy was deemed acceptable given the significant increase in practicality and ease of use [38].

The key limitation of the KR method is that it is specifically designed for and validated on white American children without growth-related pathologies. The applicability of the KR method to other ethnic groups or children with growth disorders has not been as extensively studied. Additionally, like the RWT method, the KR technique requires the availability of parental height data, which can be a practical challenge in some clinical settings. The lack of this information can limit the use of the KR method [35].

Compared to other conventional adult height prediction methods, such as Bayley-Pinneau and Tanner-Whitehouse, the KR technique has been shown to perform reasonably well. One study found that the KR method had lower prediction errors than the Bayley-Pinneau and Tanner-Whitehouse methods when applied to children with idiopathic short stature [39].

However, more recently developed machine learning-based approaches, like the Growth Curve Comparison (GCC) method, have demonstrated even greater accuracy in adult height prediction compared to the KR and other traditional techniques.

2.9.2 Novel Machine Learning Approaches

In recent years, researchers have explored the use of machine learning (ML) techniques to improve adult height prediction methods. These novel approaches aim to address the limitations of the conventional methods, such as the requirement for diverse measurements, complex calculations, and ethnic specificity. Here, we review three notable ML-based methods for adult height prediction.

2.9.2.1 BoneXpert System

The BoneXpert (BX) system is an automated method for adult height prediction that was developed to overcome the limitations of the traditional Bayley-Pinneau (BP) method [29] [40]. The BoneXpert system utilizes an automated bone age assessment algorithm to determine the child's bone age, which is then used to predict their adult height.

The BX system eliminates the need for manual bone age assessment, which is time-consuming and requires specialized expertise. Also Studies have shown that the BX method provides more accurate adult height predictions compared to the BP method, particularly for children with growth disorders or short stature [29] [40]. The BX system was developed using a diverse dataset, making it more generalizable across different populations.

Overall, the BoneXpert system represents a significant advancement in adult height prediction, providing a more objective and accurate approach that can be readily applied in clinical settings [29].

2.9.2.2 Growth Curve Comparison (GCC) Method

The Growth Curve Comparison (GCC) method is a novel ML-based approach for adult height prediction that was developed using a large cohort of over 16,000 Slovenian schoolchildren. This method uses ML algorithms to compare an individual's growth curve to a reference growth curve, allowing for more accurate predictions of adult height.

The GCC method only requires a single previous height measurement, making it a more practical and less invasive approach compared to methods that require bone age assessment or multiple measurements. Studies have shown that the GCC method outperforms other non-intrusive methods, such as the Roche-Wainer-Thissen and Khamis-Roche methods, in terms of lower prediction error and narrower confidence intervals. The GCC method was developed using a large, diverse dataset, which enhances its applicability across different populations. This has been integrated into a freely available web application, making it easily accessible for clinicians and researchers.

2.9.2.3 Spampinato et al. method

Another notable ML-based approach for adult height prediction was developed by Spampinato et al. This method utilizes deep learning techniques, specifically convolutional neural networks (CNNs), to predict adult height from a single hand radiograph.

This method is similar to the BoneXpert system, automates the bone age assessment process, eliminating the need for manual evaluation. The authors reported that their CNN-based model outperformed traditional methods, such as the Tanner-Whitehouse and Bayley-Pinneau methods, in terms of adult height prediction accuracy. The automated nature of this approach could facilitate its integration into clinical workflows, providing a more efficient and objective tool for adult height prediction. While the Spampinato et al. method is a promising development, it has not yet been as extensively validated as the BoneXpert and GCC methods in the literature.

2.9.3 Effectiveness of Automates Systems in Predicting Adult Height

The use of automated systems has been shown to improve the accuracy and consistency of adult height prediction compared to traditional manual methods [29] [41]. These systems can take into account a wider range of factors, such as ethnicity and growth disorders, to provide more personalized and accurate height predictions [29] [42].

Additionally, automated systems can process X-ray images much faster than manual methods, making them more practical for clinical use [29] [41]. This can lead to earlier detection and treatment of growth-related issues, as well as more efficient monitoring of children on growth hormone therapy or with other endocrine disorders [29] [41].

However, it is important to note that the performance of automated systems can still be influenced by factors such as image quality, patient characteristics, and the diversity of the training data used to develop the algorithms [29] [43]. Ongoing research and validation of these systems in diverse populations is necessary to ensure their widespread clinical adoption [29].

The table 3 summarizes various approaches to predicting adult height, highlighting their methodologies, strengths, and weaknesses.

Table 3: Comparison of different heigh prediction approaches

Method	Description	Advantages	Limitations
Bayley-Pinneau(BP)	Estimates adult height based on chronological age, current height, and bone age assessed from hand-wrist X-rays.	Widely used, established method.	Requires bone age assessment, uses outdated data, inaccurate for growth disorders.
Tanner-Whitehouse (TW)	Estimates adult height based on chronological age, current height, and bone age assessed from hand-wrist X-rays. Several versions (TW1, TW2, TW3).	Accounts for early/late maturers.	Requires bone age assessment, complex calculations.
Roche-Wainer-Thissen (RWT)	Estimates adult height from chronological age, current height, weight, and parental heights.	Simple and does not require bone age.	Requires parental height data, ethnically specific.
Khamis-Roche (KR)	Modified RWT method that does not require bone age assessment.	More practical than RWT, still reasonably accurate.	Limited to white American children, requires parental height.
BoneXpert (BX)	Automated bone age assessment combined with adult height prediction model.	Objective, eliminates rater variability in bone age.	Still requires bone age assessment.
Growth Curve Comparison (GCC)	Machine learning method that compares individual's growth curve to reference to predict adult height.	Non-intrusive, high accuracy, generalizable.	Requires at least one previous height measurement.
Spampinato et al.	Deep learning model that predicts adult height from a single hand radiograph.	Automated bone age assessment, potential for clinical integration.	Limited validation compared to other methods.

2.10 Summary

This chapter discussed a literature review on related research work. The following chapter will discuss the technology adapted for research and development.

Chapter 3

Technology Adapted

3.1 Introduction

This chapter contains a brief discussion on the technologies adopted during the implementation of the system. It also includes the reasons to use such technologies such as performance of the technology and suitability to the problem domain

3.2 Technologies adopted for implementation

3.2.1 Programming languages

Python was chosen as the primary programming language for implementing the bone age prediction system. Python is an object-oriented, interpreted, and flexible language that is widely used in the fields of data science, machine learning, and artificial intelligence. Its simplicity, extensive libraries, and ease of use make it an ideal choice for rapid prototyping and development of complex systems like the one proposed in this research.

3.2.2 Development environments / tools

1. **Google Colab:**

   Google Colab is a cloud-based Jupyter notebook environment that provides access to powerful hardware resources, such as GPUs and TPUs, which are essential for training deep learning models efficiently. The ability to run computationally intensive tasks on Google's infrastructure without the need for local hardware setup makes it a convenient choice for this project.

2. **Jupyter Notebook:**

   Jupyter Notebook is an interactive web-based environment that allows for the integration of code, visualizations, and documentation in a single document. It facilitates the iterative development and testing of the system's components, enabling seamless collaboration and sharing of results among the research team.

3.2.3 Libraries and frameworks:

The system utilizes several popular libraries and frameworks in Python to implement various functionalities:

1. **Data manipulation and preprocessing**:

   Libraries like **Pandas** and **NumPy** are used for efficient data handling and preprocessing of X-ray images and associated metadata. **Scikit-learn** is used for data classification, regression and analysis.

2. **Image processing**:

   The **OpenCV** (cv2) library is employed for performing image processing tasks such as filtering, smoothing, edge detection, and segmentation. These techniques are crucial for accurately extracting the carpal and phalangeal bones from the X-ray images.

3. **Data visualization**:

   Libraries like **Seaborn** and **Matplotlib** are used for creating informative visualizations to analyze the performance of the system and interpret the results.

4. **Deep learning**:

   The system leverages the **TensorFlow** and **Keras** libraries for building, training, and evaluating the convolutional neural network (CNN) models used for bone age prediction. These libraries provide a flexible and efficient framework for developing and deploying deep learning models.

3.2.4 Pre-trained Models:

1. **VGG16**:

   VGG16 is a pre-trained convolutional neural network model used for feature extraction and transfer learning. Its architecture, trained on the ImageNet dataset, allows us to leverage learned representations, improving the model's performance on our specific dataset.

2. **Roboflow API**:

   The Roboflow API is utilized for object detection, specifically to identify and extract phalangeal regions of interest (PROI) from X-ray images. This tool automates the detection process, enhancing both the efficiency and accuracy of our analysis.

3.2.5 Image Annotation Tools:

1. **Supervision (sv)**:

Supervision (sv) is used for annotating images with detections and creating masks for visualizing detected regions. This tool streamlines the annotation process, ensuring accurate markings that improve the quality of our training dataset.

3.2.6 Machine Learning Techniques:

1. **Model Training and Evaluation**:

Model training techniques were employed that included callbacks like EarlyStopping and ModelCheckpoint to optimize the training process. These methods help prevent overfitting and ensure that we retain the best-performing model weights for future use.

2. **Grad-CAM**:

Grad-CAM (Gradient-weighted Class Activation Mapping) generates attention maps that visualize the regions of an image that the model focuses on during predictions. This technique provides insights into the model's decision-making process, helping to identify which features are most influential in its predictions.

3.2.7 Version controlling:

1. **Git:**

Git, a distributed version control system, is used for managing the codebase and tracking changes throughout the development process. It enables collaboration among the research team, facilitates code sharing, and ensures the integrity and traceability of the system's implementation.

3.3 Summary

This chapter gave an overview of the technologies that will be used in the implementation of the system along with the reasons for selecting those technologies. The upcoming chapter will illustrate the approach we used in implementing the system.

Chapter 4

Approach

4.1 Introduction

This chapter outlines the approach taken to develop the automated pediatric bone age prediction system. It describes the key components of the system, the target users, the input data, the processing steps, and the expected outputs. By detailing the overall system architecture and the underlying methodology, this section provides a comprehensive understanding of how the proposed solution addresses the challenges in traditional bone age assessment methods.

4.2 System Overview

The automated pediatric bone age prediction system is designed as a hybrid solution that combines the analysis of carpal bones and phalangeal bones to provide accurate and consistent bone age assessments.

The system consists of three main modules:

- **Carpal Bone Age Assessment Module**: This module focuses on predicting the bone age of children under the age of 7 using the analysis of carpal bones in hand and wrist X-ray images.
- **Phalangeal Bone Age Assessment Module**: This module is responsible for predicting the bone age of pediatric patients using the analysis of phalangeal bones.
- **Adult Height Prediction Module**: This module takes the calculated bone ages, the patient's gender, age and current height as inputs to predict the final adult height, which is crucial for early detection of growth disorders and guiding medical interventions.

4.3 Users

The primary users of the automated pediatric bone age prediction system are healthcare professionals, including pediatricians, radiologists, and orthopedic specialists, who are responsible for assessing the growth and development of children. By providing an efficient and accurate bone age assessment tool, the system aims to support these medical practitioners in making informed decisions and delivering better patient care.

4.4 Inputs

a. The inputs for the carpal bone age assessment are displayed in table 4

Table 4: Inputs for carpal bone age assessment module

Module	Input
Data preprocessing module	Original dataset (CSV file containing labels)
Image preprocessing module	Original hand and wrist Xray images
Segmentation module	Preprocessed hand x-ray images
Hand region analysis module	Segmented hand region images
Carpal bone segmentation module	Segmented lower hand region images
Model Building and Training	Segmented carpal bones images from training and validation datasets Corresponding bone age and gender labels
Model Testing and Evaluation	Segmented carpal bones images from testing dataset Corresponding gender labels

a. The inputs for the phalangeal bone age assessment are displayed in table 5

Table 5: Inputs for phalangeal bone age assessment

Phase	Inputs
Data Loading and Preprocessing	CSV files containing training and testing labels. Paths to training and testing image datasets.
Image Processing	Images loaded from dataset paths. Parameters for image preprocessing (gamma, CLAHE, etc.). Roboflow API key for image processing.
Dataset Preparation	Preprocessed images from image processing phase. Labels extracted from CSV files.
Model Building and Training	Preprocessed images and corresponding labels. Base model (VGG16) for transfer learning. Parameters for model architecture (layers, dropout rate).
Prediction and Visualization	Trained model from model building phase. Preprocessed test images for prediction.
Grad-CAM Visualization	Trained model for Grad-CAM computation. Selected images for Grad-CAM visualization.

b. The inputs for adult height prediction module are displayed in table 6

Table 6: Inputs for adult height prediction

Module	Input
Data Preprocessing	Raw dataset including gender, chronological age (CA), bone age (BA), current height
Data Cleaning and Transformation Module	Preprocessed dataset, BP tables (advanced, retarded, average)
BP Table Lookup Module	Dataset with BA categories, BP tables
Neural Network Model Building and Training Module	Features from BP table lookup, additional engineered features, corresponding adult height labels (for training)
Model Evaluation and Validation Module	Trained model, validation dataset
Integrate Predictions	BP method predictions, NN model predictions
Prediction and Analysis Module	New input data, trained model

4.5 Process

4.5.1 Carpal bone age assessment

4.5.1.1 Data Preprocessing

The Carpal Bone Age Assessment Module preprocesses the input hand and wrist X-ray images to ensure they are suitable for analysis. The first step in the Carpal Bone Age Assessment Module is to preprocess the input data from the RSNA dataset. The process begins by filtering the dataset to include only the hand and wrist X-ray images of male patients under the age of 7 and female patients under the age of 5. This ensures that the module is focused on the appropriate age range for carpal bone age assessment. After filtering the dataset, two separate datasets are created- one for male patients and one for female patients. This allows to train and evaluate the module's performance on gender-specific data, which can improve the accuracy of the bone age predictions. Later adjusting the contrast and exposure levels is done to bring them to a consistent level. To address the significant variation in contrast and exposure levels across the images, frequency domain filtering is applied using the Fast Fourier Transform (FFT). It selectively enhances the contrast of the images, bringing them to a more consistent level. Additionally, Contrast Limited Adaptive Histogram Equalization (CLAHE) is used to

further enhance the hand region and bilateral filtering to smooth the images while preserving edges.

4.5.1.2 Hand Segmentation

After preprocessing the images, hand region segmentation from the background is done. This involves binarizing the images and identifying the largest connected component, which corresponds to the hand. This step allows to obtain a hand mask and then hand region is segmented by applying the hand mask to the original image.

4.5.1.3 Hand Alignment

To ensure consistency in the orientation of the hand images, they are aligned based on the position of the middle finger. First the hand contour is detected. Then using the contour, convex hull and convexity defects are detected. The convexity defects are identified as interdigital folds and using them fingertips are detected. Additionally, the palm center is determined by finding the largest inscribed circle within the hand contour. Using middle fingertip and calm center, the segmented hand region are being rotated such that the tip of the middle finger and the lower end of the carpal region form a straight vertical line.

4.5.1.4 Carpal Bone Region Segmentation

This sub module segments the carpal bone region by annotating the lower hand region surrounding the carpal bones. This is done by annotating the top, bottom, left and right boundaries relative to the palm center.

4.5.1.5 Carpal Bone Boundary Refinement

Adaptive canny edge detection is applied to the segmented lower hand region to obtain the edge map of the carpal Region of Interest (ROI). Local adaptive thresholding is also applied to create binary image of CROI. To tackle over-segmentation, under-segmentation, and spurious edges, integrate binary image and edge map using XOR. Edges touching the image boundary are removed and Hole filling and morphological opening are applied. Finally carpal bones are segmented.

4.5.1.6 Model Training

The Carpal Bone Age Assessment Module employs a deep learning model based on the VGG16 architecture to predict bone age from segmented carpal bone images. VGG16 is chosen for its proven effectiveness in image classification tasks, particularly due to its deep architecture and ability to learn intricate features from images.

To standardize the input, the segmented carpal bone images are resized to 224x224 pixels. The model is trained using these preprocessed images alongside a CSV file containing the corresponding bone age and gender information. By including gender as an input feature, the model is designed to capture gender-specific patterns in carpal bone development, which are crucial for accurate bone age prediction.

During training, the VGG16-based model learns to identify complex relationships between the carpal bone features and the corresponding bone ages, taking into account the gender differences. After training, the model is integrated into the overall system. When a hand and wrist X-ray image along with the patient's gender is provided, the model predicts the bone age based on the learned features.

The use of VGG16 enables the module to leverage a robust and well-established architecture, ensuring reliable and accurate predictions. The model's performance is further validated using a separate testing set to confirm its effectiveness in real-world scenarios.

4.5.2 Phalangeal bone age assessment

4.5.2.1 Data Loading and Preprocessing

Loading the CSV files containing training and testing labels is the initial step to organize data for bone age prediction. These files provide essential metadata linking image identifiers with corresponding bone ages, crucial for supervised learning tasks. Displaying the head rows of these datasets allows for an initial inspection to verify data integrity and understand its structure. Setting up the Roboflow API streamlines subsequent image processing tasks by leveraging pre-trained models and annotation tools. This integration ensures efficient inference and detection of regions of interest (ROIs) in medical images, optimizing the workflow for accurate bone age estimation.

4.5.2.2 Image Preprocessing

The preprocess image function plays a pivotal role in preparing medical images for analysis. Initially converting images to grayscale simplifies data while preserving critical anatomical details. Applying gamma correction with a value of 1.5 adjusts brightness and contrast, enhancing image clarity and feature visibility essential for precise age estimation. Contrast Limited Adaptive Histogram Equalization (CLAHE) further enhances image contrast, particularly useful in medical imaging where subtle pixel variations signify important

anatomical characteristics. Image fusion techniques blend gamma-corrected and CLAHE-enhanced images to achieve balanced brightness and contrast improvements. Fast non-local means denoising reduces artifacts, ensuring high-quality image data for subsequent analysis.

4.5.2.3 Dataset Preparation

Dataset preparation involves several essential steps to standardize and optimize data for machine learning. After preprocessing to extract and enhance regions of interest (PROIs), images are resized to a standardized dimension to ensure uniform input sizes across the dataset, necessary for training deep learning models like VGG16. Normalizing pixel values to a range between 0 and 1 standardizes intensity levels, facilitating stable and efficient model training. Splitting the dataset into training and validation sets (e.g., 80% training, 20% validation) allows for model performance evaluation on unseen data, preventing overfitting and ensuring generalization. This meticulous preparation ensures that the model learns effectively from the data, improving accuracy in predicting bone age from medical images.

4.5.2.4 Model Building and Training

Building the model involves integrating a pre-trained VGG16 base model with custom layers tailored for bone age prediction. By loading the VGG16 model pre-trained on ImageNet, foundational features are already encoded, enhancing the model's ability to extract hierarchical features from bone X-ray images. Custom layers, including global average pooling and dense layers with ReLU activation and dropout regularization (0.5), are added to fine-tune the model for specific bone age prediction tasks. The base layers of VGG16 are frozen to preserve learned features during training, optimizing computational resources and preventing overfitting. Compiled with the Adam optimizer and mean squared error loss function, the model is trained over 25 epochs on preprocessed and validated datasets. Evaluation metrics, including loss and mean absolute error (MAE), assess model performance, ensuring it accurately predicts bone age based on extracted features and annotations.

4.5.2.5 Prediction and Visualization

Prediction and visualization involve applying the trained model to predict bone age from unseen X-ray images and visualizing model insights using Grad-CAM. Post-preprocessing, including image resizing and normalization, prepares images for prediction. Utilizing Grad-CAM, the model generates heatmaps highlighting regions critical to its predictions, enhancing

interpretability and clinical relevance. These heatmaps overlay onto original images, visually correlating model decisions with anatomical features. The process facilitates a deeper understanding of how the model interprets bone age features, aiding medical professionals in decision-making and diagnosis. Through prediction and visualization, the model demonstrates its capability to provide accurate bone age estimates while offering insights into image features influencing these predictions.

4.5.3 Adult height prediction

4.5.3.1 Data Loading and Preprocessing

The Adult Height Prediction (AHP) system is designed to forecast the final adult height of children by leveraging bone age, current age, and current height data, with a particular focus on adapting the method for the Asian demographic. The dataset contains entries for bone age, chronological age, current height, and final adult height. The process begins with preprocessing and cleaning the dataset to handle missing values, outliers, and ensure data consistency. This step involves renaming columns for ease of use and standardizing units to maintain uniformity across the dataset. next step involves splitting the data into male and female subsets because male and female growth patterns differ significantly, requiring separate models for each gender.

After the initial preprocessing, bone age delay (BA_delay) is calculated by subtracting bone age from chronological age. This delay is categorized into 'Normal,' 'Delayed,' and 'Advanced' categories, based on the difference between BA and CA. (BA delay: normal BA (|CA–BA| < 1 year), advanced BA (CA - BA < -1 year), and delayed BA (CA - BA > 1 year)) These categories are essential as they help select the appropriate Bayley-Pinneau (BP) table for predicting growth potential.

The skeletal age is then converted into a format compatible with the BP tables, which are structured in specific age-month combinations. This conversion is necessary for looking up percentile adult height estimations, which are subsequently used in the neural network models. These tables provide multipliers based on the current height and bone age, which are applied to estimate adult height. To further enhance prediction accuracy and tailor the model for the Asian dataset, machine learning techniques are employed.

4.5.3.2 Model Building and Training

The neural network model is designed to predict adult height using several key features: bone age (BA), bone age delay (BA_delay), growth potential (gp), percentile maturity height (PMH),

and current height. Separate neural networks are developed for males and females due to their different growth trajectories. For the female dataset, a neural network with three hidden layers is constructed. The architecture includes layers with 128, 64, and 32 neurons, each followed by a dropout layer to prevent overfitting. The model is compiled using the Adam optimizer and Mean Squared Error (MSE) as the loss function. Training is conducted with early stopping to avoid overfitting, using a batch size of 32 and a maximum of 250 epochs. For the male dataset, a similar approach is taken, but with an additional hidden layer. This model includes layers with 256, 128, 64, and 32 neurons. Like the female model, it uses dropout layers, the Adam optimizer, and MSE as the loss function, with early stopping and a batch size of 16.

The model's performance is then validated using a portion of the dataset reserved for testing. Accuracy metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are used to evaluate the predictions. By integrating traditional BP tables with modern machine learning techniques, the proposed solution aims to enhance the accuracy of adult height predictions, making it suitable for diverse demographic groups. This approach not only leverages historical growth data but also adapts to the specific growth patterns observed in the target population.

4.6 Outputs

a. Outputs for carpal bone age assessment module are displayed in table 7

Table 7: Outputs for carpal bone age assessment

Module	Output
Data preprocessing module	Preprocessed dataset of male under the age of 7 and female under the age of 5.
Image preprocessing module	Preprocessed hand and wrist Xray images
Segmentation module	Segmented hand region
Hand region analysis module	Annotated hand region Segmented lower hand region
Carpal bone segmentation module	Segmented carpal bones
Model Building and Training	Trained VGG 16 deep learning model for carpal bone age prediction

Model Testing and Evaluation	Predicted bone age Mean Absolute Error Mean Squared Error Root Mean Squared Error R-squared Accuracy

b. Outputs for phalangeal bone age assessment are displayed in table 8

Table 8: Outputs for Phalangeal bone age assessment

Phase	Output
Data Loading and Preprocessing	Loaded training and testing labels from CSV files. Displayed head rows of training and test datasets. Initialized Roboflow API for image processing.
Image Processing	Preprocessed images: converted to grayscale, applied gamma correction, CLAHE, and noise reduction. Extracted phalangeal regions of interest (PROI) using a pre-trained model. Resized and normalized images for dataset preparation.
Dataset Preparation	Prepared training and validation sets.
Model Building and Training	Loaded VGG16 base model and added custom layers. Compiled and trained the model, evaluated with Loss and MAE metrics.
Prediction and Visualization	Predicted bone age for test dataset. Visualized prediction error distribution and actual vs. predicted bone age.
Grad-CAM Visualization	Computed and visualized Grad-CAM heatmaps for selected images.

c. Outputs for adult height prediction are displayed in table 9

Table 9: Outputs for Adult height prediction

Module	Output
Data Preprocessing	A dataset free from missing values and outliers, ready for analysis and modeling.
Data Transformation Module	Categorized dataset on gender and BA delays
Initial Height Prediction	Predicted adult height using BP tables
Neural Network Model Building	Trained neural network models ready for evaluation and prediction.
Model Evaluation and Validation Module	Detailed performance metrics and validation results indicating the effectiveness of each model.
Integrate Predictions	A single, combined prediction for adult height that leverages the strengths of each model.
Prediction and Analysis Module	Refined adult height predictions along with comprehensive analysis and visualizations to support the findings.

4.7 Summary

This chapter discussed briefly the proposed solution, module wise inputs, flow of the implementation and module wise outputs. Next chapter illustrates the design of the entire system along with the comprehensive explanation of the individual module implementations.

Chapter 5

Analysis and Design

5.1 Introduction

The analysis and design chapter discusses the design of the system that is proposed for the problem addresses in the research. It consists of the top-level architecture of the proposed system and the conceptual designs of each module in the system. It gives a description about what is done in each module and the relationship among each module.

5.2 System

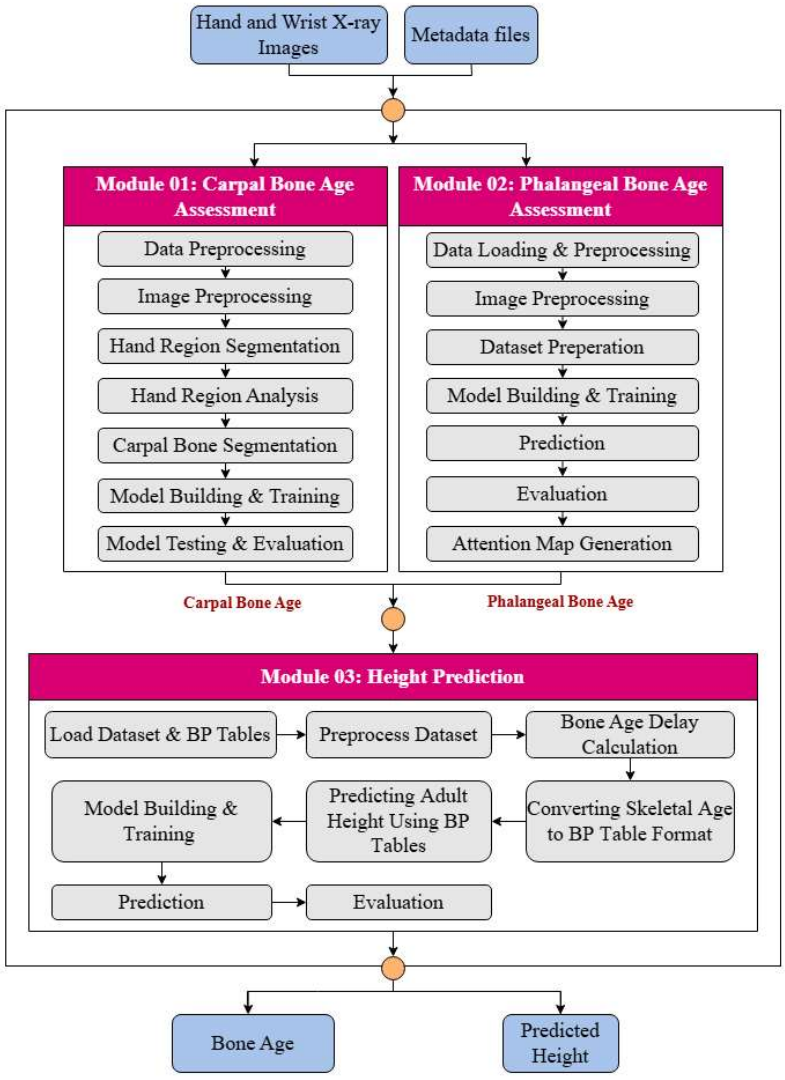


Figure 8: High-Level Review of the System

Figure 8 presents a high-level review of the system, which consists of three main modules: carpal bone age assessment, phalangeal bone age assessment, and height prediction. The process begins with the input of hand and wrist X-ray images, along with their associated metadata files. These inputs are then processed in both the carpal and phalangeal modules, where bone age is assessed using the respective carpal and phalangeal bones. Once the bone ages are determined, this information is fed into the height prediction system. The system utilizes the assessed bone ages to generate a predicted height, providing valuable insights into the growth and development of the individual. This integrated approach facilitates a comprehensive evaluation of skeletal maturity and growth potential.

5.3 Carpal bone age assessment module

The carpal bone age assessment system is a comprehensive solution that aims to accurately determine the bone age of individuals based on hand X-ray images. The process begins with the Data Preprocessing module, which filters the dataset by gender and age, and then splits the data into male and female subsets. This ensures that the subsequent analysis and model training are tailored to the specific characteristics of each gender.

The preprocessed images are then passed to the Image Preprocessing module, where various image processing techniques are applied to enhance the quality of the hand X-ray images. This includes the use of Fourier Transform (FFT), Contrast Limited Adaptive Histogram Equalization (CLAHE), bilateral filtering, and adjustments to contrast and exposure levels. These techniques help to improve the clarity and consistency of the input data, which is crucial for the subsequent segmentation and analysis steps.

The Segmentation module takes the preprocessed images and segments the hand region from the background. This is achieved by binarizing the images, identifying the largest connected component, obtaining the hand mask, and performing hand region segmentation. The segmented hand region is then passed to the Hand Region Analysis module, which detects the hand contour, identifies the convex hull and defects, detects the fingertips and palm center, and performs rotational alignment of the hand images. These insights into the hand's structure and orientation are essential for the accurate localization and analysis of the carpal bones.

The Carpal Bone Segmentation module then takes the results from the Hand Region Analysis module and segments the carpal bones by annotating their top, bottom, left, and right boundaries relative to the palm center. This precise delineation of the carpal bones is a crucial

step in the overall process, as it provides the necessary input for the subsequent model building and training.

The Model Building and Training module loads the segmented carpal bone images and their corresponding bone age labels, defines the Convolutional Neural Network (CNN) architecture, and trains separate models for male and female subjects. This gender-specific approach ensures that the models can accurately capture the nuanced differences in bone development between the two groups.

Finally, the Prediction and Analysis module allows users to input a hand X-ray image and gender, selects the appropriate model based on the gender, and predicts the bone age. This output can be used for various clinical and research applications, such as assessing growth and development, diagnosing skeletal disorders, and monitoring the progress of medical interventions.

The seamless integration of these modules, from data preprocessing to model prediction, enables a comprehensive and robust carpal bone age assessment system that can provide valuable insights and support decision-making in various healthcare and research domains.

Figure 9 outlines the high-level architecture of the carpal bone age assessment module, which encompasses seven submodules: data processing, image processing, hand segmentation, carpal bone segmentation, model building and training, and prediction and analysis. The process begins with preprocessing the input metadata and image files, followed by filtering and splitting the dataset by gender and age to obtain relevant age groups for each gender, which are then combined into a single dataset for improved accuracy. The image processing module preprocesses the data before it is fed into the hand segmentation module, which utilizes a hand mask to segment the hand and align it rotationally to ensure consistent orientation. The segmented carpal bones are then divided into training, validation, and test sets, with the training and validation sets used to build and train a model that incorporates the images, gender, and age information, and the test set employed to evaluate the model and obtain the final bone age prediction.

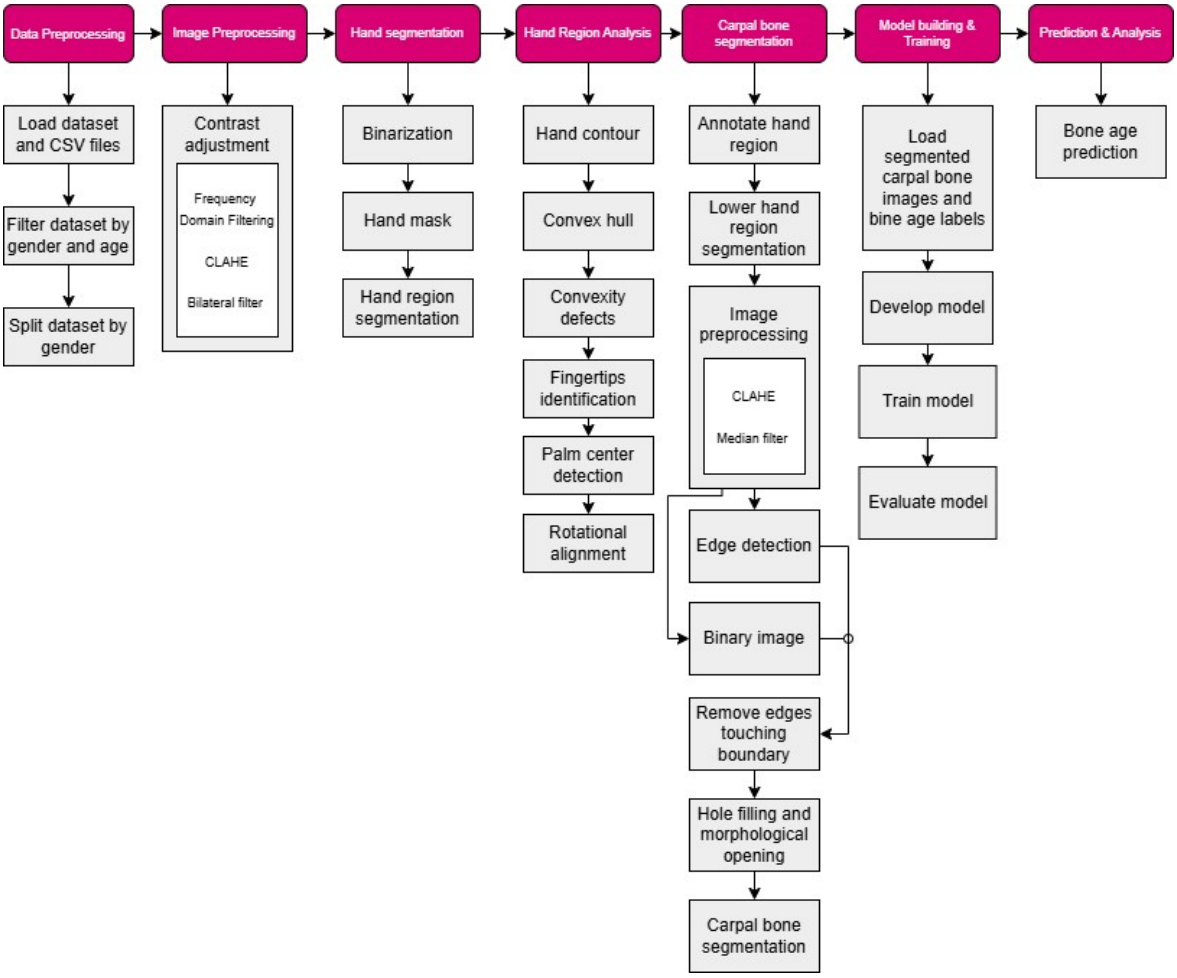


Figure 9: High level architecture for carpal bone age assessment module

5.4 Phalangeal bone age assessment module

The design of this module involves several interconnected stages, each crucial for accurate predictions. It starts with loading and preparing data from CSV files, ensuring the system has the necessary information for analysis. Next, images undergo processing to enhance their quality through techniques like grayscale conversion, gamma correction for brightness adjustment, and CLAHE for contrast improvement. This step also includes noise reduction to clean up images for better analysis.

After processing, the system focuses on identifying and isolating the phalangeal regions of interest (PROI), essential for bone age assessment. This involves using a pre-trained model to detect and annotate these regions in the images, ensuring precise extraction. The dataset is then prepared by standardizing image sizes and normalizing pixel values, making sure the data is consistent and ready for modeling.

The core of the system lies in building and training the model. Here, a base model like VGG16 is employed, enhanced with custom layers to extract relevant features from the PROI. The model is trained using the prepared dataset to minimize errors in predicting bone ages, crucially using metrics like Mean Squared Error (MSE) to gauge its accuracy.

Once trained, the model is used for predictions, estimating bone ages based on input images. Visualizing these predictions through Grad-CAM highlights the image areas influencing the model's decisions, offering insights into its reasoning. This visualization helps validate the model's accuracy and aids in its interpretation by clinicians and researchers.

Overall, this systematic approach—from data preprocessing and model construction to prediction and visualization—ensures a robust bone age prediction system. It integrates advanced image processing with deep learning techniques to provide reliable estimates, potentially impacting clinical diagnostics and advancing pediatric research.

Figure 10 illustrates the high-level architecture for the phalangeal bone age assessment module, comprising five submodules: data loading and preprocessing, image preprocessing, dataset preparation, model building and training, and prediction and validation. The first module handles data preprocessing, followed by the image preprocessing module, which processes the images to extract the phalangeal regions of interest (ROIs). In the dataset preparation module, the extracted images are resized, pixel values are normalized, and the dataset is split into training and validation sets. The subsequent module focuses on building and training the model using these training and validation sets. Finally, the prediction and visualization module generate predictions and creates attention maps, providing insights into the model's decision-making process regarding phalangeal bone age assessment.

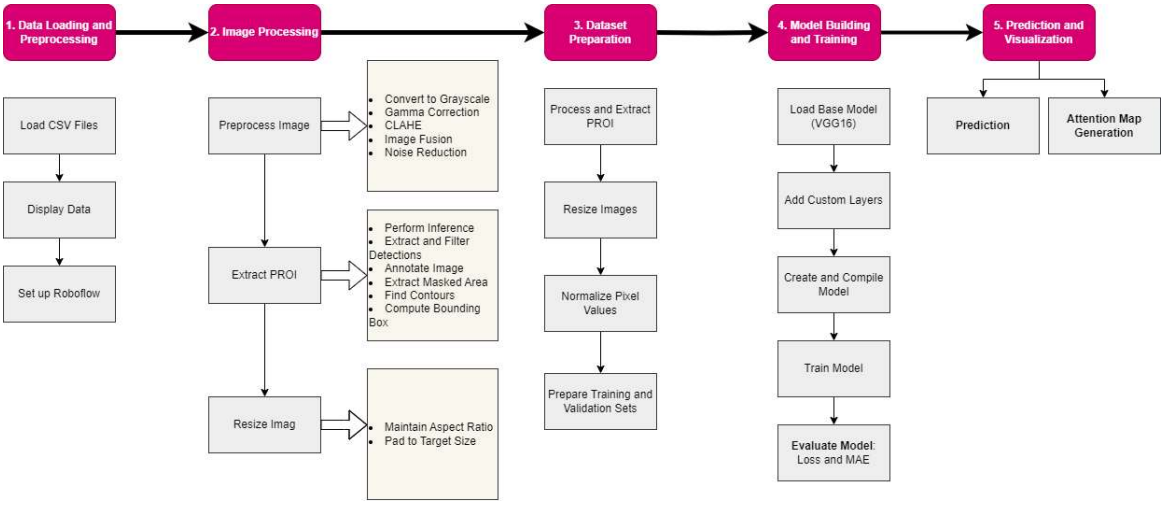


Figure 10: High level architecture for phalangeal bone age assessment module

5.5 Height prediction module

The Adult Height Prediction (AHP) system is designed to forecast the final adult height of children by leveraging bone age, current age, and current height data, with a particular focus on adapting the method for the Asian demographic. The process begins with preprocessing and cleaning the dataset to handle missing values, outliers, and ensure data consistency. This step involves renaming columns for ease of use and standardizing units to maintain uniformity across the dataset.

Following the preprocessing stage, the BA delay is calculated for each entry. BA delay, defined as the difference between chronological age (CA) and bone age (BA), is a crucial measure that helps categorize the maturity level of each child. Children are then divided into three groups based on their BA delay: normal BA ($|CA–BA| < 1$ year), advanced BA ($CA - BA < -1$ year), and delayed BA ($CA - BA > 1$ year). This categorization is essential for applying the appropriate BP tables for height prediction.

The next step involves using distinct BP tables for males and females, categorized into advanced, retarded, and average maturity, to predict adult height. These tables provide multipliers based on the current height and bone age, which are applied to estimate the adult height. To further enhance prediction accuracy and tailor the model for the Asian dataset, machine learning techniques are employed. Separate neural network models are developed for males and females, each tailored to the specific growth patterns observed in the respective gender. The models incorporate multiple features, including BA, BA_delay, gp, PMH, and current height, to predict the adult height accurately.

The evaluation module assesses the performance of the trained models using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared score. These metrics provide insights into the model's accuracy and reliability in predicting adult height. The module also compares the neural network's performance with traditional methods, highlighting the improvements achieved through the integration of machine learning. By integrating traditional BP tables with modern neural network techniques, the proposed solution aims to enhance the accuracy of adult height predictions, making it suitable for diverse demographic groups. This approach not only leverages historical growth data but also adapts to the specific growth patterns observed in the target population.

Figure 11 presents the high-level architecture for the adult height prediction module, which consists of five submodules: data preprocessing, initial height prediction, machine learning model training, integrated prediction, and prediction and visualization. The data preprocessing module takes in datasets, bone age tables, and metadata files, performing data cleaning, exploration, and splitting the dataset by gender while filtering for bone age delays. In the initial height prediction module, the Bayley-Pinneau method is utilized to calculate residuals. The subsequent module focuses on training the machine learning model, evaluating its performance, and predicting residuals based on the model. The integrated prediction module combines the predictions from both the Bayley-Pinneau method and the machine learning model to assess overall performance. Finally, the model outputs the predicted adult height, providing a comprehensive evaluation of growth potential.

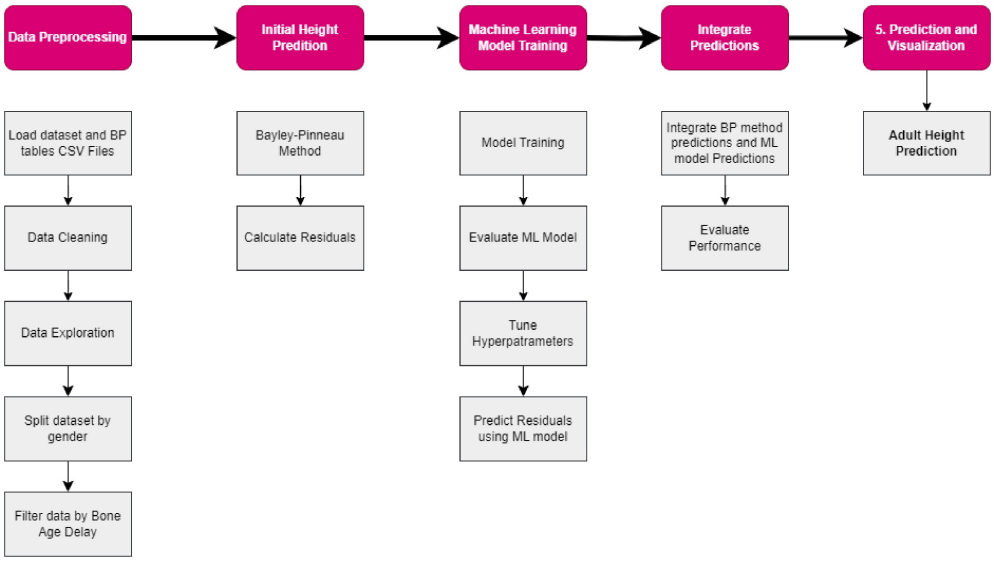


Figure 11: High level architecture for adult height prediction module

5.6 Summary

This chapter discussed the design of the solution provided for the problem addresses in the research. It included the top-level architecture of the proposed system and the workflow of each module in the system. It gave a description about what is done in each module and the relationship among modules. Next chapter will give a detailed description about how the solution is implemented

Chapter 6

Implementation

6.1 Introduction

This chapter provides the implementation details of the automated pediatric bone age prediction system, as outlined in the design phase (Chapter 5). This chapter covers the software and hardware components, the algorithms and techniques employed within each module, and the integration of these modules into the complete system. To facilitate a comprehensive understanding, this chapter includes relevant flowcharts, pseudocode, and code segments to illustrate the step-by-step processes involved in each module. Additionally, any challenges encountered during the implementation phase and the strategies adopted to overcome them are highlighted.

6.2 Carpal bone age prediction module

6.2.1 Data preprocessing

The first step in the implementation of the Carpal Bone Age Assessment Module is data acquisition. For this module RSNA (Radiological society of North America) Pediatric Bone Age Challenge dataset is used. This dataset consists of 12,814 radiological images of the left hand labeled by age and sex of patients. It has a large number of samples and has been used in several studies to develop deep learning models for bone age prediction.

As discussed in the design phase, data preprocessing begins by filtering the dataset to include only the hand and wrist X-ray images of male patients under the age of 7 and female patients under the age of 5. In order to achieve this, python's pandas library is used to read the CSV

files containing the image id, gender and bone age in months. Data is then filtered based on the patient's gender and bone age, creating two separate datasets – one for male patients and one for female patients.

The training and validation sets of the RSNA dataset are already split, so we analyze both sets to extract the relevant age limits for each gender. The process started by loading the datafiles and counting the number of data points in the training and validation sets that meet the age criteria. This step provides an understanding of the data distribution and ensures that the filtering process is working as expected.

Then the data is filtered based on the patient's gender and bone age.

- For male patients,
  - If the bone age <84 and 'male' is TRUE then store in filtered male file
- For female patients,
  - If the bone age <60 and 'male' is FALSE then store in filtered female file

Pseudocode for dataset preparation:

```
# Load the CSV files

load training dataset CSV file into df
load validation dataset CSV file into df2

# Filter the data based on gender and bone age
for training set:
filter df to get rows where boneage < 84 and male == True, store in filtered_training_males
filter df to get rows where boneage < 60 and male == False, store in filtered_training_females

for validation set:
filter df2 to get rows where boneage < 84 and male == True, store in filtered_validation_males
filter df2 to get rows where boneage < 60 and male == False, store in filtered_validation_females

# Concatenate the filtered training and validation sets
concatenate filtered_training_males and filtered_validation_males, store in filtered_all_males
concatenate filtered_training_females and filtered_validation_females, store in filtered_all_females

# Define the paths to save the CSV files
set male_csv_path to '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/filtered_all_males.csv'
set female_csv_path to '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/filtered_all_females.csv'

# Save the filtered datasets to CSV files
save filtered_all_males to male_csv_path
save filtered_all_females to female_csv_path
```

```
# Load the CSV file
df = pd.read_csv('/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/boneage-training-dataset.csv')
df2 = pd.read_csv('/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/boneage-validation-dataset.csv')
```

Figure 12: Loading datasets to data frames

```
# Counting bone ages less than 84 months (7 years) for males and less than 60 months (5 years) for females in the training set
count_training_male_less_than_84 = df[(df['boneage'] < 84) & (df['male'] == True)].shape[0]
count_training_female_less_than_60 = df[(df['boneage'] < 60) & (df['male'] == False)].shape[0]

# Counting bone ages less than 84 months (7 years) for males and less than 60 months (5 years) for females in the validation set
count_validation_male_less_than_84 = df2[(df2['boneage'] < 84) & (df2['male'] == True)].shape[0]
count_validation_female_less_than_60 = df2[(df2['boneage'] < 60) & (df2['male'] == False)].shape[0]

print("Count of bone ages in training set < 84 (males):", count_training_male_less_than_84)
print("Count of bone ages in training set < 60 (females):", count_training_female_less_than_60)
print("Count of bone ages in validation set < 84 (males):", count_validation_male_less_than_84)
print("Count of bone ages in validation set < 60 (females):", count_validation_female_less_than_60)
```

```
Count of bone ages in training set < 84 (males): 902
Count of bone ages in training set < 60 (females): 375
Count of bone ages in validation set < 84 (males): 104
Count of bone ages in validation set < 60 (females): 44
```

Figure 13: Count male and female data in the dataset

```
# Paths
training_dataset_dir = '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/carpal_training_dataset'
validation_dataset_dir = '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/validation set'
male_csv_path = '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/filtered_all_males.csv'
female_csv_path = '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/filtered_all_females.csv'
male_image_dir = '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/male_images'
female_image_dir = '/content/drive/MyDrive/FYP - Nexero/Datasets/RSNA/female_images'

# Function to ensure directory exists or recreate
def ensure_directory(directory):
    if os.path.exists(directory):
        shutil.rmtree(directory)  # Delete directory and contents
    os.makedirs(directory)  # Recreate empty directory

ensure_directory(male_image_dir)
ensure_directory(female_image_dir)

# Load CSV files
filtered_all_males = pd.read_csv(male_csv_path)
filtered_all_females = pd.read_csv(female_csv_path)

# Function to copy images based on IDs
def copy_images_based_on_id(csv_data, source_dir, destination_dir):
    for index, row in csv_data.iterrows():
        image_id = row['id']  # Assuming 'id' is the column name for image IDs
        image_path = os.path.join(source_dir, f"{image_id}.png")
        if os.path.exists(image_path):
            destination_path = os.path.join(destination_dir, f"{image_id}.png")
            shutil.copy(image_path, destination_path)
            print(f"Copied {image_id}.png to {destination_dir}")

# Copy images for males
copy_images_based_on_id(filtered_all_males, training_dataset_dir, male_image_dir)

# Copy images for females
copy_images_based_on_id(filtered_all_females, validation_dataset_dir, female_image_dir)
```

Figure 14: Filter dataset based on age and gender

Above figures 12,13,14 present the code snippets that detail the process of loading datasets into a data frame, counting male and female entries, and filtering the dataset based on age and gender.

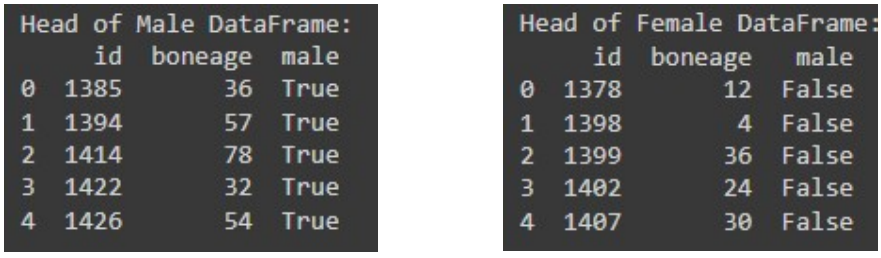
After splitting the RSNA dataset into male and female subsets based on the age limits (84 months for males, 60 months for females), statistical analysis and visualizations are performed to understand the distribution of bone ages in each gender group.

These visualizations help in understanding the distribution of bone ages in the male and female datasets separately. The histograms show the frequency of bone ages within each gender group, while the box plots provide a comparative view of the bone age distributions between genders.

The vertical lines in the plots represent the age limits used for filtering the data (84 months for males, 60 months for females). These visualizations help in verifying that the data is properly filtered and that the male and female datasets are within the desired age ranges. By analyzing the statistical distributions and visualizations, the system can gain insights into the characteristics of the male and female bone age data, which is crucial for training accurate and reliable models for bone age prediction.

Statistical analysis:

Figure 15 displays the heads of the female and male data frames, showcasing the first five entries in each dataset along with their respective IDs, bone ages, and gender. This provides a clear view of the data structure and the initial values for both groups. Meanwhile, Figure 16 presents the maximum and minimum ages within both datasets, revealing that the maximum age in the male dataset is 82 months, while the maximum age in the female dataset is 55 months. Additionally, the minimum age recorded in the male dataset is 1 month, compared to 3 months in the female dataset. These figures highlight the age distribution and range of the data, which are crucial for subsequent analyses and modeling efforts.



Head of Male DataFrame:

	id	boneage	male
0	1385	36	True
1	1394	57	True
2	1414	78	True
3	1422	32	True
4	1426	54	True

Head of Female DataFrame:

	id	boneage	male
0	1378	12	False
1	1398	4	False
2	1399	36	False
3	1402	24	False
4	1407	30	False

Figure 15: Head of male and female data frames

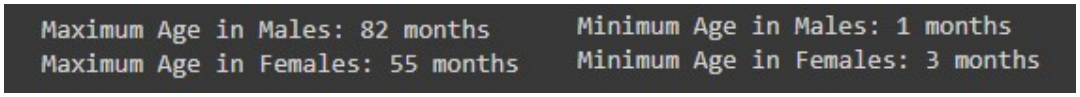


Figure 16: Maximum and minimum ages in both datasets

Visualization:

Figure 17 presents the distribution of bone ages for male and female children, with the x-axis representing bone age in months and the y-axis depicting the number of children. The male bone age distribution is skewed to the right, indicating a higher prevalence of children with advanced skeletal maturity, with a peak around 60-70 months. A vertical line at 84 months represents the upper limit for males, beyond which children are considered to have accelerated skeletal development. The female bone age distribution is also skewed to the right but to a lesser degree than males, with a peak around 40-50 months and a vertical line at 60 months denoting the female limit.

These graphs suggest that female children generally experience earlier skeletal maturation compared to males, as evidenced by the lower peak and limit for females. However, significant variability exists within each gender, emphasizing the importance of considering individual differences in bone age assessment.

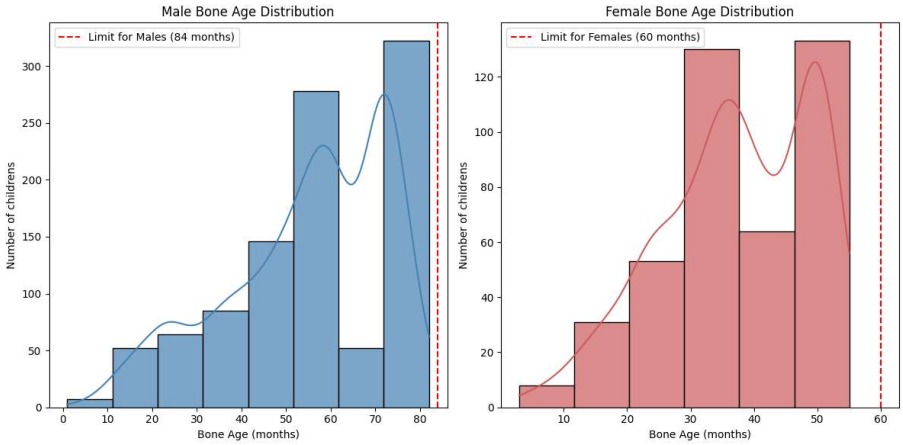


Figure 17: Distribution of bone ages for each gender

Figure 18 shows a comparison of bone age distribution between genders using a box plot. According to the plot, the median bone age is higher for males than females, indicating that male children generally have more advanced skeletal maturity. The interquartile range (IQR), which represents the middle 50% of the data, is slightly larger for males, suggesting greater variability in bone age among male children. Notably, there are no outliers in either group. The vertical lines in the plot represent the limits for male and female children, demonstrating that a larger proportion of male children have bone ages exceeding the limit compared to female children. These findings align with the observations made from the histograms in Figure 17,

further confirming that female children tend to exhibit lower bone ages than male children, while male children display more variability in their skeletal maturity.

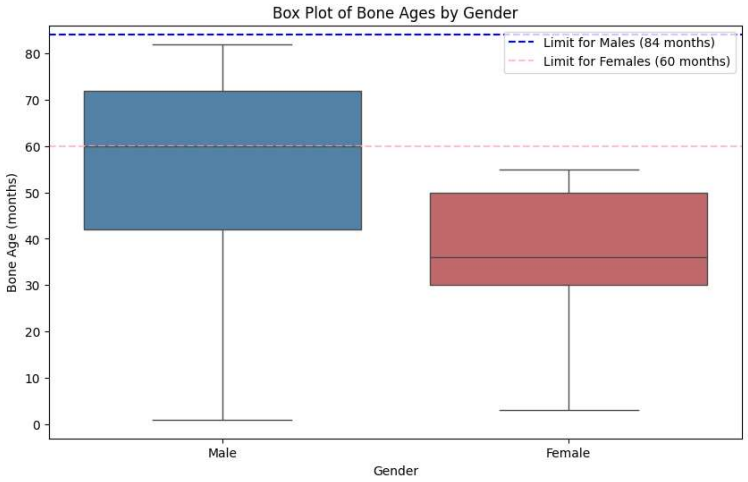


Figure 18: Compare the bone age distributions between genders

6.2.2 Image preprocessing

The image preprocessing phase is a crucial step in the Carpal Bone Age Assessment Module, as it ensures the input images are enhanced and optimized for the subsequent analysis and bone age prediction tasks.

Frequency domain filtering

The first step of the image preprocessing pipeline is applying frequency domain filtering to enhance the contrast of the input images. This step is particularly important in this project, as the hand and wrist X-ray images often exhibit significant variations in contrast and exposure levels due to differences in imaging equipment, patient positioning, and other factors.

Conventional image enhancement techniques, like histogram equalization, may not be sufficient to address these challenges, as they often fail to selectively amplify the high-frequency components associated with the edges and details of the anatomical structures of interest.

To overcome this limitation, the system employs a frequency domain filtering approach, which leverages the principles of Fourier analysis to selectively enhance the contrast of the input images.

Compared to traditional image enhancement techniques, the frequency domain filtering method offers several advantages: such as allowing for selective enhancement of high-

frequency components which are associated with the edges and details of the anatomical structures of interest, such as the carpal bones, able to mitigate the effects of varying imaging conditions and bring the input images to a more uniform level of contrast, improving the overall quality and consistency of the data and improve the performance of the subsequent segmentation and bone age prediction tasks, leading to more accurate and reliable results.

Pseudocode for frequency domain filtering function:

```
#frequency domain filtering function
def frequency_domain_filtering(input_image):
    # Step 1: Convert the input image to grayscale
    grayscale_image = convert_to_grayscale(input_image)
    # Step 2: Apply Fast Fourier Transform (FFT) to the grayscale image
    frequency_domain_image = apply_fft(grayscale_image)
    # Step 3: Shift the frequency domain representation to the center
    shifted_frequency_domain_image = shift_frequency_domain(frequency_domain_image)
    # Step 4: Create a high-pass filter mask
    high_pass_filter_mask = create_high_pass_filter_mask(shifted_frequency_domain_image)
    # Step 5: Apply the high-pass filter mask to the frequency domain representation
    filtered_frequency_domain_image = apply_high_pass_filter(shifted_frequency_domain_image,
high_pass_filter_mask)
    # Step 6: Perform inverse FFT to convert the filtered image back to the spatial domain
    filtered_spatial_domain_image = apply_inverse_fft(filtered_frequency_domain_image)
    # Step 7: Calculate the magnitude of the complex-valued inverse FFT
    magnitude_image = calculate_magnitude(filtered_spatial_domain_image)
    # Step 8: Normalize the resulting image to the range of 0 to 255
    normalized_image = normalize_image(magnitude_image)
    return normalized_image
```

Frequency domain filtering starts by grayscale conversion. The input color image is converted to grayscale using the cv2.cvtColor function. This simplifies the subsequent processing steps and reduces the computational complexity.

The grayscale image is then transformed to the frequency domain using the Fast Fourier Transform (FFT) implemented by the cv2.dft function. This step converts the spatial domain representation of the image into the frequency domain, where different frequency components can be selectively manipulated.

The frequency domain representation is shifted to the center of the image using the np.fft.fftshift function. This step ensures that the low-frequency components are located at

the center of the frequency domain image, while the high-frequency components are at the periphery.

A high-pass filter mask is created by defining a circular region around the center of the frequency domain image. The pixels within this circular region are set to 1, while the rest are set to 0. This mask selectively amplifies the high-frequency components, which are often associated with the edges and details of the hand and carpal bone regions.

The high-pass filter mask is applied to the frequency domain representation by multiplying it with the shifted FFT. This step effectively enhances the high-frequency components while suppressing the low-frequency components.

Inverse FFT is performed using the cv2.idft function to convert the filtered image back to the spatial domain. The magnitude of the complex-valued inverse FFT is then calculated using the cv2.magnitude function.

The resulting image is normalized to the range of 0 to 255 using the cv2.normalize function, ensuring the pixel values are within a suitable range for further processing.

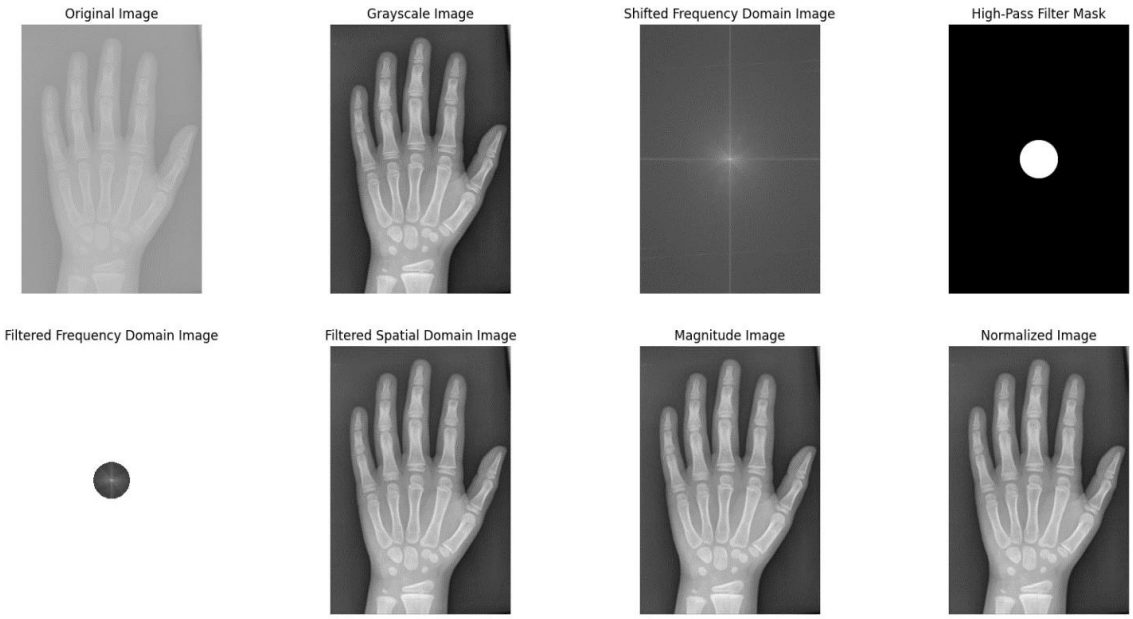


Figure 19: Outputs of Frequency domain filtering

The frequency domain filtering process is visually depicted in Figure 19, showcasing the transformation of the original image into the final normalized image. This comprehensive illustration includes the intermediate outputs, such as a grayscale image, a shifted frequency domain image, a high-pass filter mask, a filtered frequency domain mask, a filtered spatial

domain mask, and a magnitude image. Each stage of the process is clearly presented, allowing for a thorough understanding of how the original image is modified through the application of frequency domain filtering techniques. This figure serves as a valuable tool for visualizing the impact of each step on the image, ultimately leading to the enhanced, normalized image that emphasizes specific features while reducing noise and irrelevant details. The sequential display of these outputs provides a clear and informative representation of the frequency domain filtering process.

The histogram analysis in figure 20 reveals a significant difference before and after the adjustment of frequency domain filtering. The adjusted histogram is more spread out than the original, showcasing a wider range of intensity values. The peak of the histogram is no longer as sharp and has shifted slightly to the right, indicating that the adjustment has increased the intensity values of many pixels. Additionally, the overall shape of the adjusted histogram is less skewed to the right compared to the original, suggesting a reduction in the contrast of the image. Overall, these changes imply that the adjustment has brightened the image while reducing its contrast, likely by increasing the intensity values of darker pixels and decreasing those of brighter pixels, leading to a more balanced representation of intensity values.

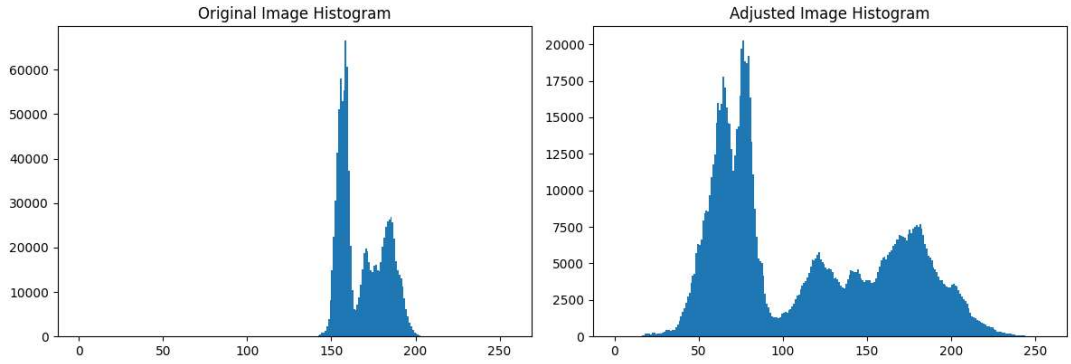


Figure 20: Histogram of original image before and after FDT

The code presented in Figure 21 outlines the implementation of frequency domain filtering, incorporating several essential functions for image processing. It includes functions to convert an image to grayscale, apply the Fast Fourier Transform (FFT), and shift the frequency domain representation to the center for better visualization. Additionally, the code demonstrates how to create a high-pass filter mask and apply this mask to filter the frequency domain representation. Following this, the inverse FFT is performed to convert the filtered data back to the spatial domain. The magnitude of the resulting image is calculated, and the final step involves normalizing the image to enhance its visual quality. This comprehensive code

structure provides a step-by-step approach to frequency domain filtering, detailing all necessary functions and processes involved in transforming the original image into a filtered and normalized output.

```python
# Function to convert to grayscale
def convert_to_grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Function to apply FFT
def apply_fft(image):
    return cv2.dft(np.float32(image), flags=cv2.DFT_COMPLEX_OUTPUT)

# Function to shift the frequency domain representation to the center
def shift_frequency_domain(dft):
    return np.fft.fftshift(dft)

# Function to create a high-pass filter mask
def create_high_pass_filter_mask(dft_shift, radius=90):
    rows, cols = dft_shift.shape[:2]
    crow, ccol = rows // 2, cols // 2
    mask = np.zeros((rows, cols, 2), np.uint8)
    center = [crow, ccol]
    x, y = np.ogrid[:rows, :cols]
    mask_area = (x - center[0]) ** 2 + (y - center[1]) ** 2 <= radius*radius
    mask[mask_area] = 1
    return mask

# Function to apply the high-pass filter mask
def apply_high_pass_filter(dft_shift, mask):
    return dft_shift * mask

# Function to perform inverse FFT
def apply_inverse_fft(fshift):
    f_ishift = np.fft.ifftshift(fshift)
    return cv2.idft(f_ishift)
```

```python
# Function to calculate magnitude
def calculate_magnitude(img_back):
    return cv2.magnitude(img_back[:, :, 0], img_back[:, :, 1])

# Function to normalize image
def normalize_image(image):
    return cv2.normalize(image, None, 0, 255, cv2.NORM_MINMAX)
```

```python
# Step-by-step processing
grayscale_image = convert_to_grayscale(original_image)
frequency_domain_image = apply_fft(grayscale_image)
shifted_frequency_domain_image = shift_frequency_domain(frequency_domain_image)
high_pass_filter_mask = create_high_pass_filter_mask(shifted_frequency_domain_image)
filtered_frequency_domain_image = apply_high_pass_filter(shifted_frequency_domain_image, high_pass_filter_mask)
filtered_spatial_domain_image = apply_inverse_fft(filtered_frequency_domain_image)
magnitude_image = calculate_magnitude(filtered_spatial_domain_image)
normalized_image = normalize_image(magnitude_image)
```

Figure 21: Frequency domain filtering code

CLAHE (Contrast Limited Adaptive Histogram Equalization)

After applying frequency domain filtering to enhance the contrast of the input X-ray images, the next step in the image preprocessing pipeline is to apply Contrast Limited Adaptive Histogram Equalization (CLAHE) and bilateral filtering for further enhancement.

CLAHE is a technique that enhances the local contrast of the image by applying histogram equalization to small, non-overlapping regions of the image. By applying CLAHE after the frequency domain filtering, the system can further improve the visibility and prominence of

the hand and carpal bone regions, making them more suitable for subsequent segmentation and analysis tasks.

First the input image, which is frequency domain filtered image is divided into small, non-overlapping regions called tiles. Then for each tile, a histogram is calculated, and the cumulative distribution function (CDF) is computed. The CDF is then used to redistribute the pixel values within each tile, effectively enhancing the local contrast. In order to avoid amplifying noise in homogeneous regions, a clip limit is applied to the histogram. This limits the maximum number of pixels that can be mapped to a single bin in the histogram. Finally, the processed tiles are combined using bilinear interpolation to eliminate artificial boundaries between tiles.

Bilateral filtering

After applying CLAHE, a bilateral filter is used to reduce noise while preserving edges and important anatomical features. The bilateral filter is a non-linear, edge-preserving filter that combines pixel values based on both their spatial proximity and their intensity similarity.

For each pixel in the CLAHE enhanced image, a window is defined around the pixel. Within the window, a weight is assigned to each neighboring pixel based on both its spatial distance from the center pixel and its intensity difference from the center pixel. The weighted average of the neighboring pixels is calculated and assigned as the new value for the center pixel. This process is repeated for all pixels in the image, smoothing the image and removing unwanted artifacts, such as those caused by the bones and skin.

Pseudocode for apply CLAHE and bilateral filter:

```
# Create a CLAHE object (Contrast Limited Adaptive Histogram Equalization)
clahe = create CLAHE object with clipLimit=2 and tileGridSize=(3, 3)
# Apply CLAHE
clahe_equalized_image = apply clahe.apply to gray_adjusted_image
# Apply Bilateral Filter after CLAHE
equalized_image = apply cv2.bilateralFilter to clahe_equalized_image with d=9, sigmaColor=75, sigmaSpace=75
```

Figure 22: Output results of CLAHE and biliteral filtering

The output of applying Contrast Limited Adaptive Histogram Equalization (CLAHE) and bilateral filtering is demonstrated in Figure 22, which includes the original image, the image after CLAHE adjustment, and the final image after bilateral filtering. The original image serves as the baseline, while the application of CLAHE enhances the contrast by redistributing pixel intensity values, resulting in a more balanced and visually appealing image. Following this, the bilateral filtering process is applied to smooth the image while preserving edges, further improving the overall quality. This sequence effectively illustrates how these image processing techniques can enhance visual clarity and detail, making the final output more suitable for analysis or presentation.

6.2.3 Segmentation module

The next step in the image preprocessing pipeline is the segmentation module, which starts with binarization and following mask generation and hand region segmentation.

Binarization

Binarization is the process of converting a grayscale image into a binary image, where each pixel is either black (0) or white (1). This step is crucial for the subsequent segmentation of the hand and carpal bone regions in the X-ray images. By applying binary thresholding, the system can isolate the hand and carpal bone regions from the background in the preprocessed X-ray images. This step helps to simplify the subsequent segmentation tasks, where the relevant anatomical structures will be identified and extracted for further analysis.

In this step, binary thresholding is applied to each filtered image using the threshold value 70 and the maximum value set to 250.

The choice of the threshold value (70 in this case) is an important parameter that can be adjusted based on the characteristics of the input images. Experimenting with different threshold values and evaluating the results is often necessary to find the optimal setting for the specific dataset and application.

The code snippets in Figure 23 demonstrate the process of applying binary thresholding to each equalized image. This technique is crucial for segmenting the images by converting them into binary format, where pixels are classified as either foreground or background based on a specified threshold value. Following the application of binary thresholding, Figure 24 showcases several examples of the binarized images. These images highlight the effectiveness of the thresholding process, as key features become more distinct and easier to analyze. The transformation from equalized images to binarized outputs illustrates the enhancement of important structures within the images, facilitating further analysis and interpretation.

```
# Apply binary thresholding to each image in equalized_images
for equalized_image, filename in equalized_images:
    _, binary_image = cv2.threshold(equalized_image,70, 250, cv2.THRESH_BINARY)
    binary_threshold_images.append((binary_image, filename))
```

Figure 23: Binarization code

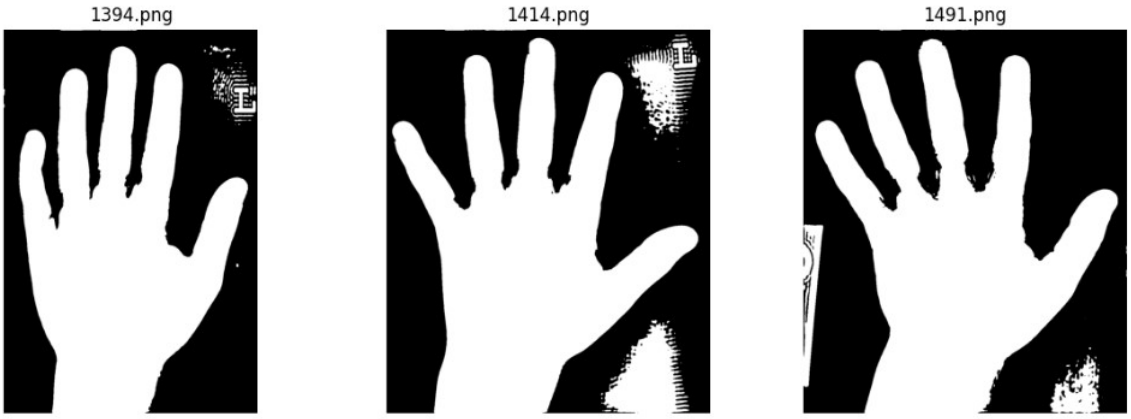


Figure 24: Output results of binarization

Mask generation

The mask generation step aims to identify and extract the relevant anatomical structures, specifically the hand and carpal bone regions, from the binarized images. This is achieved by applying a connected component analysis and selecting the largest component as the segmented mask.

In this step, all the connected components in the binary images will be identified using "cv2.connectedComponents". Then identify the largest component by iterating through the labels and finding the one with the maximum size. Following that, a mask is created for the largest component by setting pixels belonging to the largest component to 255 (white) and the rest to 0 (black). Additionally, Check if the mask is connected to the border of the image. If so, remove a two-pixel border from the edges of the mask to ensure the mask is not touching the image boundaries. This last step is crucial for future stages to make hand region to be isolated from the image boundaries.

Figure 25: Output results of generated hand masks

Figure 25 displays several images with masks applied, illustrating the results of the segmentation process. These masked images highlight the regions of interest identified in the previous steps, providing a clearer view of the significant features within the images. Complementing this, Figure 26 presents code snippets that detail the process of finding all connected components within the masked images, identifying the largest connected component, and creating a mask specifically for that component. The code also includes a crucial check to determine if the mask is connected to the image border, as this connection would prevent the application of contours to the mask. If the mask is found to be connected to the border, the code removes a 2-pixel border from the edges of the mask to ensure that contouring can be effectively applied. This systematic approach enhances the accuracy of the segmentation and prepares the data for subsequent analysis.

```python
# Initialize a list to store the segmented masks
mask_images = []

# Define a function to segment the largest component and create a mask
def segment_largest_component(binary_image):
    # Find all connected components (8-connectivity)
    num_labels, labels_im = cv2.connectedComponents(binary_image)

    # Find the largest component (excluding background)
    max_label = 1
    max_size = 0
    for label in range(1, num_labels):
        size = np.sum(labels_im == label)
        if size > max_size:
            max_size = size
            max_label = label

    # Create a mask for the largest component
    mask = (labels_im == max_label).astype(np.uint8) * 255

    # Check if the mask is connected to the border
    if (mask[0, :].any() or mask[-1, :].any() or mask[:, 0].any() or mask[:, -1].any()):
        # Remove a two-pixel border from the edges of the mask
        mask[0:2, :] = 0
        mask[-2:, :] = 0
        mask[:, 0:2] = 0
        mask[:, -2:] = 0

    return mask
```

Figure 26: Code for hand mask generation

6.2.4 Hand region segmentation

The final step in the segmentation module is to apply the masks to the original X-ray images and extract the hand regions.

Figure 27 presents the output of the segmented hand region image, showcasing the effectiveness of the applied mask in isolating the entire hand from the original image. This segmentation highlights the relevant anatomical features, making it easier for further analysis or processing. Complementing this output, Figure 28 displays the code snippets that include the function used to apply the previously generated mask to the original image. This function effectively segments the hand region by utilizing the mask to extract only the relevant portion of the image, ensuring that the surrounding background is excluded. Together, these figures illustrate the successful implementation of image processing techniques to achieve precise segmentation of the hand region, laying the groundwork for subsequent analysis and evaluation.

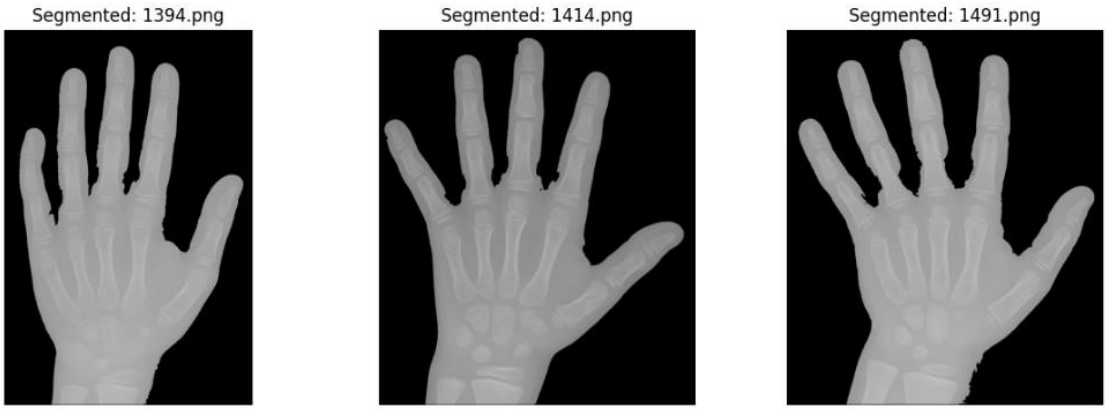


Figure 27: Output for segmented images

```
# Function to apply the mask to the original image
def apply_mask(original_image, mask):
    # Ensure the mask is binary
    mask = mask // 255
    # Apply mask to original image
    segmented_image = original_image * np.dstack([mask, mask, mask])
    return segmented_image
```

Figure 28: Code for segmenting hand regions

Hand region analysis

The hand region analysis module involves analyzing the segmented hand region and performing rotational alignment and lower hand region segmentation based on those analyses.

Contour and Convex Hull detection

The first step in the hand region analysis involves contour detection and convex hull detection. The segmented hand region mask is converted to a binary image, where the foreground pixels (hand region) are represented as 1 and the background pixels are represented as 0. Then the binary mask is dilated using a small kernel to eliminate any self-intersections that may be present in the mask. This helps to smooth the contours and ensure they accurately represent the hand region.

The cv2.findContours function is used to detect the contours in the binary mask and these contours represent the boundaries of the hand region as displayed in figure 29.

```
# Apply dilation to eliminate self-intersections
kernel = np.ones((3, 3), np.uint8)
segmented_mask = cv2.dilate(segmented_mask, kernel, iterations=1)

# Find contours
contours, _ = cv2.findContours(segmented_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Figure 29: Code for finding contours

Once the contours are detected, for each contour the cv2.convexHull function is used to find the convex hull as presented in figure 30. The convex hull represents the smallest convex polygon that contains the contour.

```
# Draw convex hull in blue
hull_points = cv2.convexHull(approx_contour)
cv2.drawContours(mask_rgb, [hull_points], -1, (255, 0, 0), 2)
```

Figure 30: Code for drawing convex hull

Convexity defects detection

The cv2.convexityDefects function is then used to detect the convexity defects, which are the points where the contour deviates from the convex hull. Each convexity defect is represented by a list containing the following attributes: Start point, Endpoint, Farthest point Approximate distance to the farthest point.

For each convexity defect, the angle at the farthest point is calculated using the cosine theorem. The formula used is:

$$a^2 = b^2 + c^2 - 2bc\cos A$$

Where a, b, and c are the lengths of the sides of the triangle formed by the start point, endpoint, and farthest point.

The angle A is then calculated as:

$$A = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2bc}\right)$$

If the calculated angle A is less than 90 degrees, the point is considered a convexity defect. These defect points represent the interdigital folds between the fingers. The start and end points of the convexity defects are considered to be the fingertips. These points are stored in the finger_points list, ensuring that each unique point is only stored once.

Figure 31 presents the code snippet for convexity defect detection, a crucial step in identifying the fingers and their tips within the segmented hand region. This code utilizes the previously generated hand mask and contours to calculate the convex hull and detect any convexity defects. Complementing the code, Figure 32 showcases the output images resulting from this process. These images include:

1. The binary hand region image
2. The contour around the hand region, highlighted in green
3. The convex hull around the hand region, shown in red
4. The fingertips, marked with green points
5. The convexity defects, indicated by blue points in between two fingers

This comprehensive output effectively visualizes the key features extracted from the segmented hand region, enabling the identification of individual fingers and their tips.

```python
# Find convexity defects
if len(hull) > 2:
    defects = cv2.convexityDefects(approx_contour, hull)

    if defects is not None:
        for i in range(defects.shape[0]):
            s, e, f, d = defects[i, 0]
            start = tuple(approx_contour[s][0])
            end = tuple(approx_contour[e][0])
            far = tuple(approx_contour[f][0])

            # Only store unique start and end points
            if start not in finger_points:
                finger_points.append(start)
            if end not in finger_points:
                finger_points.append(end)

            # Calculate angle at defect point
            a = np.linalg.norm(np.array(start) - np.array(end))
            b = np.linalg.norm(np.array(start) - np.array(far))
            c = np.linalg.norm(np.array(end) - np.array(far))

            angle = np.degrees(np.arccos((b**2 + c**2 - a**2) / (2 * b * c)))

            # Mark defect if the angle is less than 90 degrees
            if angle < 90.0:
                cv2.circle(mask_rgb, far, 5, (0, 0, 255), -1)
                cv2.circle(mask_rgb, start, 5, (0, 255, 255), -1)
                cv2.circle(mask_rgb, end, 5, (0, 255, 255), -1)
```

Figure 31: Code for finding convexity defects

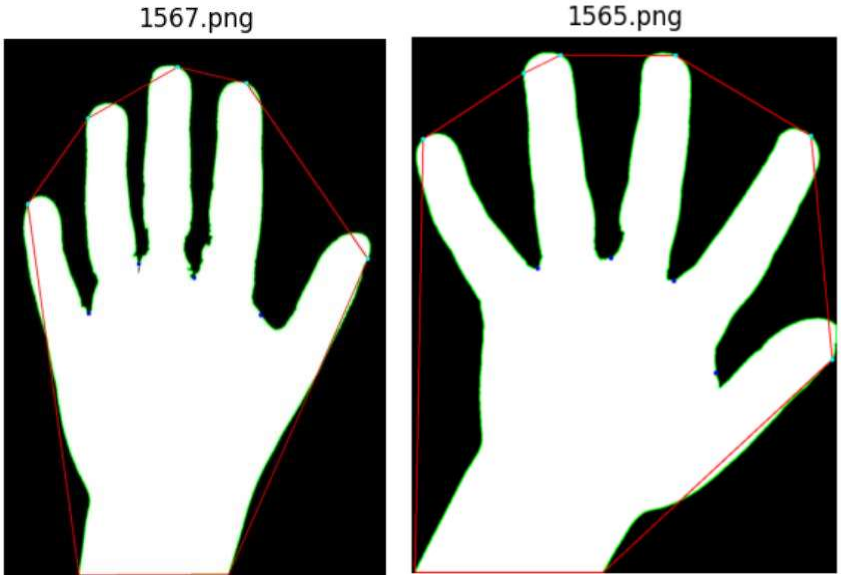


Figure 32: Output results of contour, convex hull and convexity defects

 # Step 1: Binarization

- Convert the mask to a binary image

 # Step 2: Dilation

- Apply a dilation operation to the binary mask using a small kernel

 # Step 3: Contour Detection

- Use the cv2.findContours function to detect the contours in the binary mask

 # Step 3: Convex Hull detection

- Iterate through the detected contours

- For each contour, use the cv2.convexHull function to calculate the convex hull

- The convex hull represents the smallest convex polygon that contains the contour

 # Step 4: Convexity Defect Detection

- Iterate through the detected contours again

- For each contour, use the cv2.convexityDefects function to detect the convexity defects

- The convexity defects represent the points where the contour deviates from the convex hull

 # Step 5: Finger Point Identification

- For each convexity defect, extract the start point, endpoint, and farthest point

- Calculate the angle at the farthest point using the cosine theorem

- If the calculated angle is less than 90 degrees, consider the farthest point as a convexity defect (interdigital fold)

- Store the start and end points (fingertips) in a list, ensuring that each unique point is only stored once

Palm center detection

The next step in the hand region analysis pipeline is the palm center detection, which involves identifying the center of the palm region in the segmented hand images. It detects the largest inscribed circle within the hand contour and marks its center.

After identifying the largest contour, which is the hand region, using the cv2.distanceTransform function distance transform of the binary image will be identified. Then locate the maximum value in the distance transform, which corresponds to the center of the largest inscribed circle and draw the largest inscribed circle and its center on the image.

The code snippets for finding the palm center are shown in Figure 33, detailing the algorithm used to locate the palm's central point within the segmented hand region. The output images in Figure 34 illustrate this process, with the palm area highlighted by green circles and the palm center marked by a red point. This visualization effectively demonstrates the accuracy of the palm center detection, providing a clear representation of its location within the hand region for further analysis.

```
# Function to detect the largest inscribed circle within the hand contour and mark its center
def detect_palm_circle(image):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    _, thresh = cv2.threshold(gray, 1, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    max_contour = max(contours, key=cv2.contourArea)

    # Find the largest inscribed circle
    dist_transform = cv2.distanceTransform(thresh, cv2.DIST_L2, 5)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(dist_transform)

    # Draw the largest inscribed circle and its center
    result_image = image.copy()
    cv2.circle(result_image, max_loc, int(max_val), (0, 255, 0), 2)  # Draw the circle
    cv2.circle(result_image, max_loc, 5, (255, 0, 0), -1)  # Draw the center as a small dot
    palm_middle_point.append({"filename": filename, "middle_point": max_loc})
    print(f"Filename: {filename}")
    print("middle_point:", max_loc)

    return result_image, max_loc, int(max_val)
```

Figure 33: Code for finding palm center

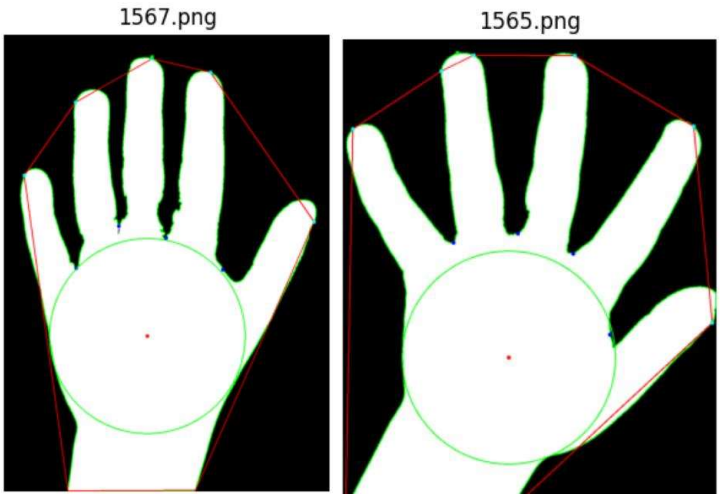


Figure 34: Output images for palm center detected images

Rotational alignment

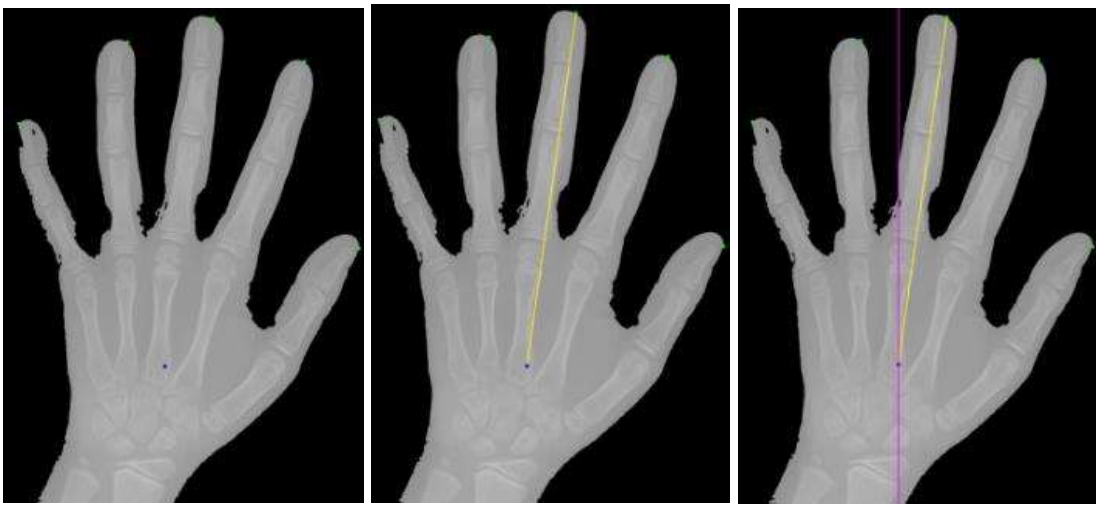
The hand rotational alignment step is a crucial part of the image preprocessing pipeline in the Carpal Bone Age Assessment Module. This step ensures that the hand region in the X-ray images is properly aligned, with the fingertips pointing upwards.

The first step is to identify the finger points and the palm center for each segmented hand region image. This information was obtained in the previous step of the pipeline, where the contour detection, convex hull analysis, and convexity defect detection were performed.

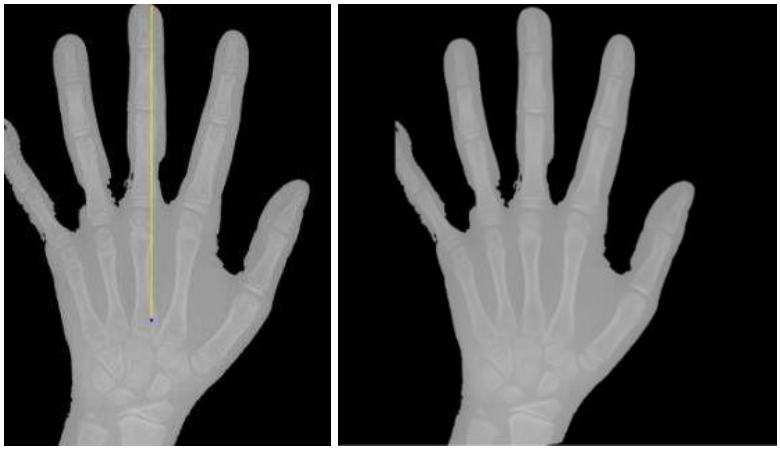
From the identified finger points, the middle finger point is determined by sorting the points by their x-coordinate in descending order and selecting the middle point. This middle finger point is used as a reference for the rotational alignment.

Then a line is drawn from the middle finger point to the palm center point. This line represents the orientation of the hand. Following that, the angle between the line from the middle finger to the palm center and the vertical line through the palm center is calculated. This angle represents the degree of rotation required to align the hand.

If the calculated angle is not zero, the image is rotated by the negative of the calculated angle, using the palm center as the rotation center. This effectively aligns the hand region with the vertical line, ensuring that the fingertips are pointing upwards.



(a) *(b)* *(c)*



(d) *(e)*

Figure 35: Output results of hand rotation steps

Figure 35 displays the output images of the rotational alignment process. In image (a), the fingertips and palm center are marked on the segmented hand region. Image (b) connects the palm center to the middle finger with a yellow line, while image (c) features a pink line drawn vertically through the palm center. The angle between these lines is used to rotate the hand so that this angle becomes 0, ensuring the fingertips point upwards. The rotated image with the points and lines is shown in (d), while image (e) presents the final output, displaying the rotated hand without any annotations. This sequence effectively illustrates the steps taken to achieve proper alignment of the hand for further analysis.

Figures 36, 37, and 38 present the code snippets utilized for the processes involved in the rotational alignment of the hand. These snippets detail the specific algorithms and functions

implemented to identify the fingertips and palm center, calculate the necessary rotation angle, and apply the rotation to the segmented hand image.

```python
# Find finger points and palm middle point for this image
finger_points = find_finger_points(image_filename, fingertips)
middle_point = find_middle_point(image_filename, palm_middle_point)
# Adjust finger points by removing outliers in y-coordinate
if finger_points:
    y_coordinates = [point[1] for point in finger_points]  # Extract y-coordinates
    y_median = np.median(y_coordinates)   # Calculate median of y-coordinates
    y_diff = np.abs(np.array(y_coordinates) - y_median)  # Calculate absolute difference from median
    y_threshold = 500  # Define threshold for outlier removal
    # Remove points with y-values exceeding the threshold
    adjusted_finger_points = [finger_points[i] for i in range(len(finger_points)) if y_diff[i] < y_threshold]
    # Sort adjusted finger points by x-coordinate in descending order
    sorted_finger_points = sorted(adjusted_finger_points, key=lambda point: point[0], reverse=True)
    middle_index = len(sorted_finger_points) // 2 # Calculate index of the middle finger point
    if len(sorted_finger_points) % 2 == 0:  # Even number of points
        middle_index -= 1
    middle_finger_point = sorted_finger_points[middle_index]  # Get the middle finger point
    cv2.circle(hand_region_image, middle_finger_point, 5, (0, 0, 255), -1)  # Draw the Blue circle for middle finger point
    # Draw lines from the middle finger point to the palm center point
    if middle_point:
        cv2.line(hand_region_image, middle_finger_point, middle_point, (0, 255, 255), 2)  # Yellow line
    # Draw all finger points
    for point in adjusted_finger_points:
        cv2.circle(hand_region_image, point, 5, (0, 255, 0), -1)  # Green circles for finger points
if middle_point:
    cv2.circle(hand_region_image, middle_point, 5, (255, 0, 0), -1)  # Red circle for palm middle point
    # Draw a vertical line through the palm center
    height = hand_region_image.shape[0]
    cv2.line(hand_region_image, (middle_point[0], 0), (middle_point[0], height), (255, 0, 255), 2)  # Pink/cyan line
```

Figure 36: Code for drawing lines for rotation alignment

```python
# Calculate the angle between the vertical line and the line from palm center to middle finger point
delta_y = middle_finger_point[1] - middle_point[1]
delta_x = middle_finger_point[0] - middle_point[0]
angle = np.degrees(np.arctan2(delta_x, delta_y))  # Angle in degrees

# Check if the line is vertically straight
if angle != 0:
    if angle > 90:
        angle -= 180
    elif angle < -90:
        angle += 180
    save_image = rotate_and_center_image(save_image, -angle, middle_point)
    hand_region_image = rotate_and_center_image(hand_region_image, -angle, middle_point)

    # Save the rotated image
    output_path = os.path.join(output_dir, image_filename)
    cv2.imwrite(output_path, save_image)
```

Figure 37: Code for rotating the image

```python
# Define a function to resize an image while maintaining aspect ratio and padding to 512x512
def resize_image_with_padding(image_path, target_size=512):
    image = cv2.imread(image_path)
    height, width = image.shape[:2]
    aspect_ratio = width / height if height != 0 else 1

    if aspect_ratio > 1:
        new_width = target_size
        new_height = int(target_size / aspect_ratio)
    else:
        new_height = target_size
        new_width = int(target_size * aspect_ratio)

    resized_image = cv2.resize(image, (new_width, new_height))
    canvas = np.zeros((target_size, target_size, 3), dtype=np.uint8)
    x_offset = (target_size - new_width) // 2
    y_offset = (target_size - new_height) // 2
    canvas[y_offset:y_offset + new_height, x_offset:x_offset + new_width] = resized_image
    return canvas
```

Figure 38: Code for resizing the image

Lower hand region annotation

The final step in the hand region analysis module involves segmenting the lower hand region by drawing lines relative to the palm center. This process ensures that the lower hand region is accurately identified and isolated for further analysis.

The palm center is identified by finding the centroid of the largest contour in the hand mask. The centroid is calculated using the moments of the contour, which provides the center of mass of the hand region.

The carpal upper border is estimated based on the centroid and the height of the hand. A percentage of the hand height is added to the centroid's y-coordinate to determine the estimated upper border. This ensures that the upper border is below the centroid, effectively segmenting the lower hand region.

As illustrated in figure 39 following lines are drawn on the image to segment the lower hand region:

1. **Largest Contour**: The largest contour of the hand region is drawn in green to outline the hand and wrist.
2. **Palm Center**: The centroid of the hand region is marked with a red circle to indicate the palm center.
3. **Carpal Upper Border**: A line is drawn from the left edge of the image to the estimated carpal upper border, and another line is drawn from the right edge of the image to the same point. This line represents the upper border of the carpal bones.
4. **Vertical Lines**: Two vertical lines are drawn, one on the left side of the image and another on the right side. These lines are positioned based on the centroid and the image width, ensuring they are offset from the centroid by a specified percentage.
5. **New Horizontal Line**: A new horizontal line is drawn at a specified percentage of the image height, below the carpal upper border. This line helps to further segment the lower hand region.

The code for generating the lines that annotate the lower hand region is shown in Figure 40. This snippet outlines the specific functions and parameters used to draw the lines on the image, effectively marking key areas of interest in the lower hand region.

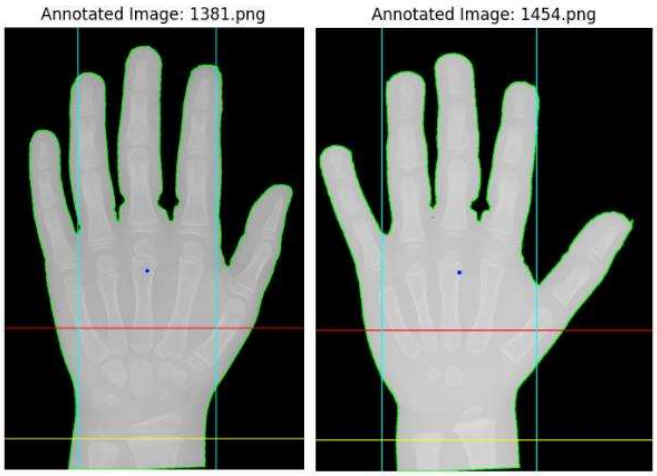


Figure 39: Output for annotated hand regions

```
# Find contours in the hand mask
contours, _ = cv2.findContours(hand_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Ensure at least one contour was found
if contours:
    # Find the largest contour (hand and wrist)
    largest_contour = max(contours, key=cv2.contourArea)
    # Calculate area of the largest contour
    area = cv2.contourArea(largest_contour)
    # Calculate centroid (center of mass) of the largest contour
    M = cv2.moments(largest_contour)
    centroid_x = int(M['m10'] / M['m00'])
    centroid_y = int(M['m01'] / M['m00'])
    # Estimate carpal upper border based on centroid and hand height
    hand_height = image.shape[0]  # Assuming the height of the image is the hand's vertical extent
    carpal_upper_border_percent = 0.12
    estimated_carpal_upper_border = centroid_y + int(hand_height * carpal_upper_border_percent)  # Ensure it's below the centroid
    # Draw centroid and upper border on the image
    cv2.drawContours(image, [largest_contour], 0, (0, 255, 0), 2)  # Draw largest contour
    cv2.circle(image, (centroid_x, centroid_y), 5, (255, 0, 0), -1)  # Draw centroid
    cv2.line(image, (0, estimated_carpal_upper_border), (image.shape[1], estimated_carpal_upper_border), (0, 0, 255), 2)  # Draw upper border
    # Calculate vertical line positions based on centroid
    vertical_line_offset_percent = 0.2
    left_vertical_line_x = int(centroid_x - image.shape[1] * vertical_line_offset_percent)
    right_vertical_line_x = int(centroid_x + image.shape[1] * vertical_line_offset_percent)
    # Draw vertical lines
    cv2.line(image, (left_vertical_line_x, 0), (left_vertical_line_x, image.shape[0]), (255, 255, 0), 2)
    cv2.line(image, (right_vertical_line_x, 0), (right_vertical_line_x, image.shape[0]), (255, 255, 0), 2)
    # Define the position of the new horizontal line (e.g., 15% above the bottom of the image)
    lower_horizontal_line_percent = 0.88
    lower_horizontal_line_y = int(hand_height * lower_horizontal_line_percent)
    # Draw the new horizontal line
    cv2.line(image, (0, lower_horizontal_line_y), (image.shape[1], lower_horizontal_line_y), (0, 255, 255), 2)  # New line in yellow
    # Save the annotated image with contours, centroid, upper border, vertical lines, and new horizontal line
    annotated_filename = os.path.join(output_dir, 'annotated_' + image_filename)
    cv2.imwrite(annotated_filename, image)
    # Append annotated image for later display
    annotated_images.append((image_filename, image))
```

Figure 40: Code for hand region annotation

6.2.5 Carpal bone segmentation module

The region below the estimated carpal upper border, bounded by the left and right vertical lines, is extracted and saved as the segmented lower hand regions as illustrated in figure 41. This region represents the area of interest for the subsequent carpal bone segmentation and analysis.

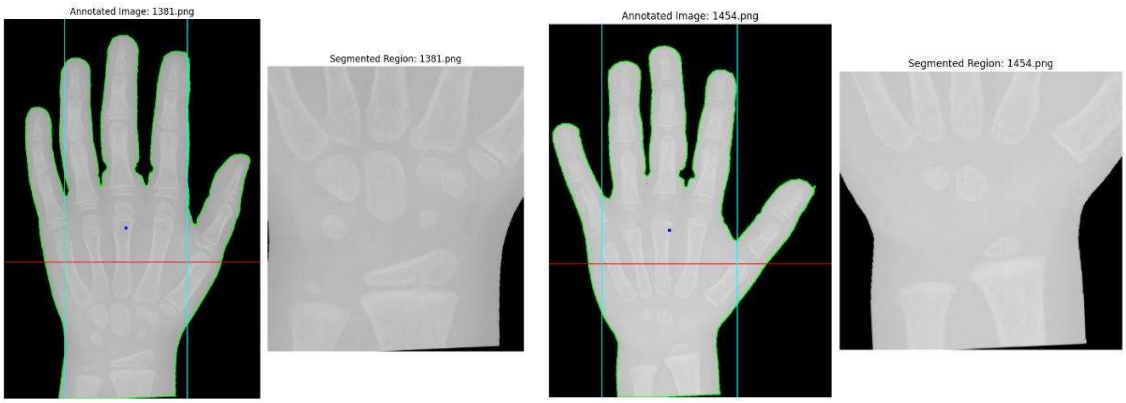


Figure 41: Output for segmented lower hand regions

As the final step, image preprocessing techniques are applied to enhance the image. The segmented lower hand region is first processed using CLAHE to enhance the local contrast of the image. This helps to improve the visibility of the carpal bones and other anatomical structures. Then a median blur filter is then applied to the CLAHE-processed image. This helps to reduce noise and smooth the image while preserving edges.

Then adaptive Canny edge detection is applied to the blurred image and adaptive Canny edge detection is also applied to the blurred image. These techniques use a dynamic threshold, which adapts to the local image characteristics, to detect the edges more accurately.

The binary image and the edge map obtained from the adaptive Canny edge detection are combined using a bitwise XOR operation. This step helps to retain the edges that are not connected to the image borders, which are likely to correspond to the carpal bones.

Any edges that touch the border of the image are removed, as they are less likely to be related to the carpal bones and may introduce unwanted artifacts.

Hole filling and morphological opening operations are performed on the processed image to fill any gaps within the carpal bone regions and smooth the boundaries.

The resulting image, after the above processing steps, represents the segmented carpal bone regions within the lower hand region.

Figure 42 illustrates the outputs for carpal bone segmentation, starting with the segmented lower hand region and progressing through several key steps. The process begins with a CLAHE-enhanced image, which improves contrast for better visibility of structures. Next, the enhanced image is binarized, highlighting relevant features, followed by the application of Canny edge detection to identify edges within the image. A combined image then showcases

the results of both binarization and edge detection, emphasizing the key structures. The filled image further clarifies the areas of interest by filling in gaps, leading to the final output that presents the carpal bone mask. Furthermore. Figure 43 displays the code used for these steps.

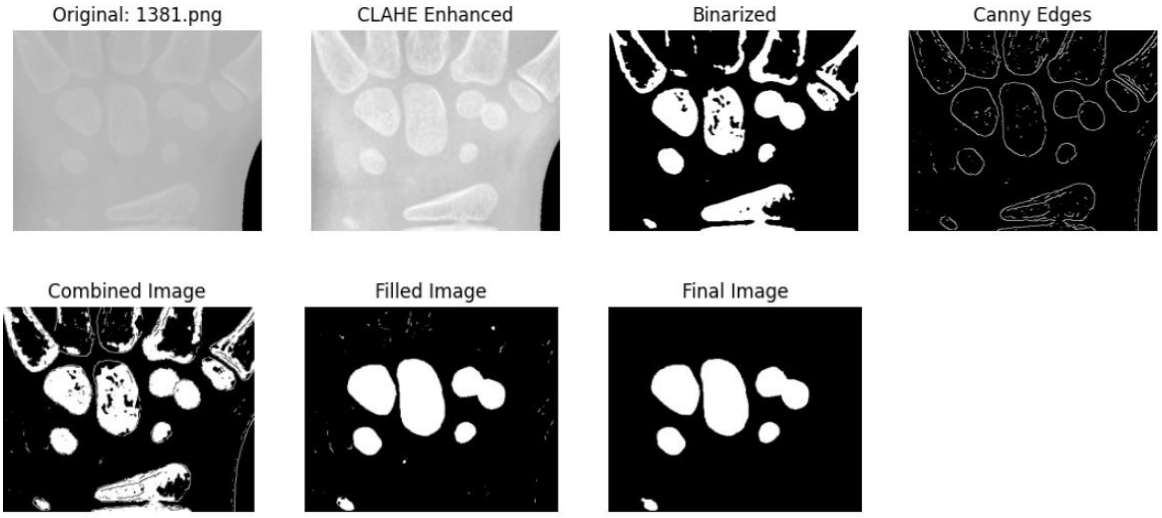


Figure 42: Outputs for carpal bone segmentation

```
# Carpal bone segmentation function
def process_images():
    # Step 1: Equalize and binarize the image
    equalized_image, smoothed_image, binary_image = equalize_and_binarize(image, threshold_value=58)

    # Step 2: Detect and mark circular or elliptical objects
    marked_image, enhanced_image, edges_image = detect_and_mark_objects(image)

    # Step 3: Create a combined image by XORing the binary and edge-detected images
    combined_image = combine_images(binary_image, edges_image)

    # Step 4: Load the corresponding filled image from the filled images directory
    filled_image = load_filled_image(filled_images_dir, filename)

    # Step 5: Remove small objects from the filled image
    if filled_image_exists(filled_image):
      final_image = remove_small_objects(filled_image, min_area=100)
```

```python
# Function to detect and mark circular or elliptical objects in images
def detect_and_mark_objects(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(5, 5))
    enhanced = clahe.apply(gray)
    blurred = cv2.medianBlur(enhanced, 5)
    edges = cv2.Canny(blurred, 30, 30)
    contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    result = image.copy()
    for contour in contours:
        if len(contour) >= 5:
            ellipse = cv2.fitEllipse(contour)
            center, axes, orientation = ellipse
            major_axis, minor_axis = axes
            aspect_ratio = major_axis / minor_axis
            if 0.8 < aspect_ratio < 1.2:
                cv2.ellipse(result, ellipse, (0, 255, 0), 2)
    return result, enhanced, edges

# Function to equalize and binarize the image
def equalize_and_binarize(image, threshold_value=58):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    equalized_image = cv2.equalizeHist(gray_image)
    smoothed_image = cv2.bilateralFilter(equalized_image, d=7, sigmaColor=75, sigmaSpace=75)
    _, binary_image = cv2.threshold(smoothed_image, threshold_value, 255, cv2.THRESH_BINARY)
    return equalized_image, smoothed_image, binary_image

# Function to fill holes in the binary image
def fill_holes(binary_image):
    inverted_binary = cv2.bitwise_not(binary_image)
    h, w = binary_image.shape
    mask = np.zeros((h+2, w+2), np.uint8)
    cv2.floodFill(inverted_binary, mask, (0, 0), 255)
    filled_image = cv2.bitwise_not(inverted_binary)
    return filled_image

# Function to fill holes in the combined image and remove border-touching objects
def fill_surrounded_holes(combined_image):
    kernel = np.ones((3, 3), np.uint8)
    dilated = cv2.dilate(combined_image, kernel, iterations=1)
    filled_image = fill_holes(dilated)
    eroded = cv2.erode(filled_image, kernel, iterations=1)
    h, w = eroded.shape
    contours, _ = cv2.findContours(eroded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours:
        x, y, width, height = cv2.boundingRect(cnt)
        if x == 0 or y == 0 or (x + width) == w or (y + height) == h:
            cv2.drawContours(eroded, [cnt], -1, 0, thickness=cv2.FILLED)
    return eroded
```

```python
# Function to remove small noise and contours
def remove_small_objects(image, min_area=150):
    kernel = np.ones((3, 3), np.uint8)
    opened = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel, iterations=2)
    contours, _ = cv2.findContours(opened, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    mask = np.zeros_like(image)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area >= min_area:
            cv2.drawContours(mask, [cnt], -1, 255, thickness=cv2.FILLED)
    filtered_image = cv2.bitwise_and(opened, mask)
    return filtered_image
```

Figure 43: Code for carpal bone segmentation

6.2.6 Model Building and Training

The Carpal Bone Age Prediction module is the core component of the overall system, responsible for accurately predicting the bone age of children based on their hand X-ray images. This module utilizes a Convolutional Neural Network (CNN) model to learn the relevant features from the segmented hand and carpal bone regions and make the final bone age prediction.

Model building:

The CNN model for the Carpal Bone Age Prediction module consists of the following key components:

1. **Input Layer**: The input to the model is the segmented hand region image, which is preprocessed and normalized to a fixed size of 224x224 pixels. This resolution aligns with the input requirements of the VGG16 model and ensures consistency in image dimensions.
2. **VGG16 Base Model**: The VGG16 model, pre-trained on the ImageNet dataset, is used. Its convolutional layers extract hierarchical features from the input images.
    I. **Convolutional Layers:** The model includes multiple convolutional layers that progressively extract higher-level features from the input images.
    II. **Pooling Layers:** MaxPooling layers are employed between convolutional layers to reduce the spatial dimensions and computational load.
    III. **Flatten Layer:** The final output of the VGG16 base is flattened to form a single long feature vector.
3. **Custom Dense Layers**:
    I. **Dense Layer 1:** A fully connected layer with 256 neurons, followed by a Dropout layer to prevent overfitting.
    II. **Dense Layer 2:** Another fully connected layer with 128 neurons, followed by another Dropout layer.
4. **Gender Input**: A separate input branch for gender information, which is processed through a dense layer before being combined with the main image feature vector.

5. **Concatenate Layer**: The outputs of the dense layers and the gender input layer are concatenated, creating a combined feature representation.

6. **Output Layer**: The final dense layer predicts the bone age as a single continuous value.

Training Process:

To effectively train the model, segmented carpal bones images were resized to 224x224 pixels as displayed in figure 44. This size was chosen to match the VGG16 input requirements, balancing computational efficiency with the retention of critical image features.

```
def resize_image_with_aspect_ratio(image_path, target_size=TARGET_SIZE):
    with Image.open(image_path) as img:
        # Calculate the new size maintaining the aspect ratio
        img.thumbnail(target_size, Image.ANTIALIAS)

        # Create a new image with the target size and black padding
        new_img = Image.new("RGB", target_size, (0, 0, 0))  # Black background
        offset = ((target_size[0] - img.width) // 2, (target_size[1] - img.height) // 2)
        new_img.paste(img, offset)

        return new_img
```

Figure 44: Code for image resizing

Post-resizing, data augmentation was applied to the training dataset to enhance the model's robustness. Techniques such as rotation, width and height shifting, shearing, zooming, and horizontal flipping were employed. These augmentations simulate various orientations and perspectives of carpal bone X-rays, helping to reduce overfitting and improve the model's ability to generalize. Figure 45 presents the code snippet that applies various augmentation techniques to the images.

```
# Image data generator for augmentation
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = datagen.flow(X_train, y_train, batch_size=32)
```

Figure 45: Code for data augmentation

The VGG16 model, pre-trained on ImageNet, was utilized for feature extraction due to its proven efficacy in image classification. The convolutional base was frozen during the initial training phase to leverage pre-learned filters and reduce computational requirements.

Custom layers were added on top of VGG16 to adapt the model for bone age prediction. After feature extraction, the flattened output was processed through fully connected layers with 256 and 128 neurons, respectively, with dropout layers to address overfitting. Gender information was included as an additional input, processed through a dense layer, and concatenated with the features from the VGG16 model. This incorporation of gender data allows the model to account for biological differences in bone maturation.

The model was compiled using the Adam optimizer with a learning rate of 0.001, focusing on minimizing mean squared error (MSE), which is standard for regression tasks like bone age prediction. Mean Absolute Error (MAE) was also monitored to assess performance.

Training involved several techniques to ensure optimal performance:

- Model Checkpoints: Save the best-performing model based on validation loss.
- Early Stopping: Halt training when validation loss shows no significant improvement, preventing overfitting and saving computational resources.
- Reduce Learning Rate on Plateau: Adjust the learning rate when validation loss plateaus to allow finer adjustments in later epochs.

A combined generator fed both image data and gender information into the model during training. Validation was performed on a separate dataset to evaluate the model's generalization ability. The training process was closely monitored using loss and MAE metrics on both training and validation datasets to ensure effective learning and prediction accuracy.

Figure 46 showcases the code for building the VGG16 model, detailing the addition of custom layers and the freezing of certain layers to retain pre-trained weights. It illustrates how inputs are incorporated into the model, the completion of the model architecture, and the implementation of checkpoints and early stopping mechanisms to optimize training. Following this, Figure 47 displays the model training code, which runs for 30 epochs, providing a comprehensive view of the training process. Figure 48 presents the results from the first 10 epochs of model training, including metrics such as training loss, mean absolute error (MAE), validation loss, validation MAE, and learning rate, offering insights into the model's performance during the initial training stages.

```python
# VGG16 base model
input_shape = X_train.shape[1:]
vgg = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)

# Freeze the layers except the last 4 layers
for layer in vgg.layers[:-4]:
    layer.trainable = False

# Add custom layers
x = vgg.output
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)

# Add gender input
gender_input = Input(shape=(1,), name='gender_input')
gender_dense = Dense(16, activation='relu')(gender_input)

# Combine VGG and gender inputs
combined = concatenate([x, gender_dense])
age_output = Dense(1, name='age_output')(combined)

model = Model(inputs=[vgg.input, gender_input], outputs=age_output)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse', metrics=['mae'])

# Model Checkpoint
checkpoint = ModelCheckpoint('/content/drive/MyDrive/FYP - Nexero/BAA model/VGG16_bone_age_model.h5',
                             monitor='val_loss', save_best_only=True, mode='min')

# Early Stopping
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Reduce Learning Rate on Plateau
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001)

# Create combined generator
def combined_generator(image_gen, gender_input):
    while True:
        batch_img, batch_age = image_gen.next()
        gender_data = gender_input[:len(batch_img)]
        yield [batch_img, gender_data], batch_age
```

Figure 46: Code for building CNN model

```python
# Train the model
history = model.fit(
    combined_generator(train_generator, genders_train),
    steps_per_epoch=len(train_generator),
    validation_data=([X_val / 255.0, genders_val], y_val),
    epochs=30,
    callbacks=[checkpoint, early_stop, reduce_lr]
)
```

Figure 47: Code for model training

```
Epoch 1/30
36/36 [==============================] - ETA: 0s - loss: 3977.4373 - mae: 31.0457 /usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:
  saving_api.save_model(
36/36 [==============================] - 1039s 29s/step - loss: 3977.4373 - mae: 31.0457 - val_loss: 213.8960 - val_mae: 11.8832 - lr: 0.0010
Epoch 2/30
36/36 [==============================] - 1035s 29s/step - loss: 455.9202 - mae: 17.0578 - val_loss: 179.3186 - val_mae: 10.4991 - lr: 0.0010
Epoch 3/30
36/36 [==============================] - 1026s 29s/step - loss: 424.0126 - mae: 16.4569 - val_loss: 172.2455 - val_mae: 10.4237 - lr: 0.0010
Epoch 4/30
36/36 [==============================] - 1031s 29s/step - loss: 363.8234 - mae: 15.2219 - val_loss: 226.6202 - val_mae: 12.2016 - lr: 0.0010
Epoch 5/30
36/36 [==============================] - 1022s 29s/step - loss: 341.1684 - mae: 14.7102 - val_loss: 176.5676 - val_mae: 10.7565 - lr: 0.0010
Epoch 6/30
36/36 [==============================] - 1023s 29s/step - loss: 298.0867 - mae: 13.6932 - val_loss: 212.4650 - val_mae: 11.6153 - lr: 0.0010
Epoch 7/30
36/36 [==============================] - 1023s 29s/step - loss: 289.9188 - mae: 13.7056 - val_loss: 239.8715 - val_mae: 12.4022 - lr: 0.0010
Epoch 8/30
36/36 [==============================] - 1022s 29s/step - loss: 304.9752 - mae: 13.8306 - val_loss: 211.0992 - val_mae: 11.7112 - lr: 0.0010
Epoch 9/30
36/36 [==============================] - 1022s 28s/step - loss: 284.9716 - mae: 13.2489 - val_loss: 296.0063 - val_mae: 13.8395 - lr: 2.0000e-04
Epoch 10/30
36/36 [==============================] - 1024s 29s/step - loss: 264.8679 - mae: 12.9636 - val_loss: 214.2299 - val_mae: 11.6403 - lr: 2.0000e-04
```

Figure 48: Model training (first 10 epoch)

As illustrated in two-line plot figures 49 both the training and validation loss decrease over the epochs, indicating that the model is effectively learning from the data. However, a slight gap between the training and validation loss suggests a potential overfitting issue, where the model performs better on the training data than on unseen validation data. Similarly, the training and validation mean absolute error (MAE) also show a decreasing trend, with a noticeable gap between the two, further hinting at overfitting.

The observed overfitting can be attributed to the inadequacies in the dataset used for training the model. Specifically, the dataset lacks a continuous representation of all ages, resulting in certain age groups having no images, while others may have multiple images. This inconsistency can lead to model learning patterns that are not representative of the entire population, ultimately affecting its ability to generalize. Additionally, the relatively small size of the dataset, consisting of only around 2,000 images, is insufficient for a robust image analysis project. A larger and more diverse dataset is essential to provide the model with a comprehensive understanding of the variations in the data, thereby reducing the risk of overfitting and improving its performance on unseen data.

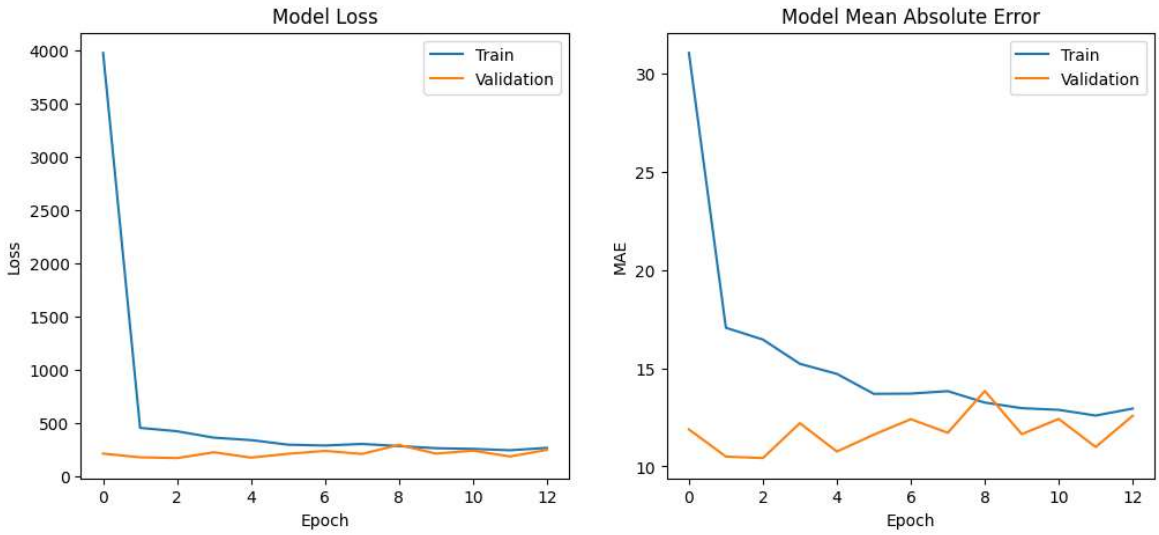


Figure 49: Plot training & validation loss values

6.2.7 Model Testing

The model testing for the carpal bone age assessment system involved a series of detailed steps to evaluate the performance of the trained model using unseen test data.

Firstly, the pre-trained model was loaded from the specified path. This model, built on the VGG16 architecture, was used to predict bone ages from the test images.

Test images were prepared by resizing them to 224x224 pixels and normalizing pixel values to the range [0, 1]. Gender information was also processed and included as part of the input to the model.

The model generated predictions for the bone ages based on these preprocessed images. The code for testing model is represented in figure 50.

```
# Function to load and preprocess images
def load_and_preprocess_image(filepath):
    img = image.load_img(filepath, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0  # Normalize to [0, 1]
    return img_array

# Prepare test images and gender information
test_images = []
test_genders = []
test_boneages = []

for index, row in test_df.iterrows():
    img_path = os.path.join(resized_dir, str(row['id']) + '.png')  # Convert id to string
    test_images.append(load_and_preprocess_image(img_path))
    test_genders.append(1 if row['male'] else 0)  # Convert boolean to numeric (1 for male, 0 for female)
    test_boneages.append(row['boneage'])

# Stack images and convert lists to numpy arrays
test_images = np.vstack(test_images)
test_genders = np.array(test_genders).reshape(-1, 1)  # Ensure gender data is in the correct shape
test_boneages = np.array(test_boneages)

# Prepare the inputs for the model
test_images_with_gender = [test_images, test_genders]

# Predict bone age using the model
predicted_boneages = model.predict(test_images_with_gender)
```

Figure 50: Code for model testing

6.3 Phalangeal bone age prediction module

6.3.1 Data Loading and Processing

This dataset includes separate directories for training and test images, essential for model training and evaluation. The CSV files containing training and test labels are loaded to provide detailed metadata associated with each image, crucial for supervised learning tasks.

To enhance dataset management and preprocessing capabilities, integration with the Roboflow API plays a pivotal role. Utilizing an API key facilitates seamless interaction with the Roboflow platform, known for its robust support in image dataset management. This integration allows for efficient augmentation, version control, and model integration, streamlining the preprocessing pipeline and ensuring data quality and consistency.

Exploratory data analysis begins with displaying the initial rows of both the training and test datasets. This step provides an overview of the dataset's structure, helping to understand label distributions and identifying any initial preprocessing needs. Such insights are critical for formulating effective preprocessing strategies and ensuring dataset readiness for subsequent modeling tasks.

Furthermore, establishing connectivity with specific projects within the Roboflow workspace, such as "x-ray-mask," enhances the implementation phase. This project-centric approach facilitates access to specialized models tailored for tasks like object detection or segmentation in medical imaging. Leveraging these models through Roboflow supports advanced image processing techniques, vital for extracting and analyzing features relevant to bone age prediction.

Figure 51 shows the code snippet for libraries and figure 52 shows data loading and preprocessing, detailing the steps involved in importing the dataset, applying necessary transformations, and preparing the data for use in the model training and evaluation processes.

```
import os
import pandas as pd
import cv2
import numpy as np
import tensorflow as tf
import supervision as sv
from roboflow import Roboflow
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import matplotlib.pyplot as plt
```

Figure 51: Code for Importing Libraries

```
# Define the paths to the datasets
data_path = 'rsna-bone-age-dataset'
train_image_path = os.path.join(data_path, 'boneage-training-dataset')
test_image_path = os.path.join(data_path, 'boneage-test-dataset')

# Load the CSV files containing the labels
train_labels = pd.read_csv(os.path.join(data_path, 'boneage-training-dataset.csv'))
test_labels = pd.read_csv(os.path.join(data_path, 'boneage-test-dataset.csv'))

# Display the first few rows of the training and test datasets
print("Training Dataset:")
print(train_labels.head())
print("\nTest Dataset:")
print(test_labels.head())

# Initialize the Roboflow API
api_key = "VgBMG3qxNXBxCvDKJfkT"
rf = Roboflow(api_key=api_key)
project = rf.workspace().project("x-ray-mask")
roboflow_model = project.version(1).model
```

Figure 52: Code for Data Loading and Preprocessing

6.3.2 Image Preprocessing and Dataset Preparation

The preprocessing of X-ray images begins with converting the image into grayscale, which simplifies subsequent analysis by reducing the color complexity to a single intensity channel. Following this, gamma correction is applied to adjust the brightness and contrast of the image, enhancing visibility of anatomical details crucial for bone age assessment. Adaptive Histogram Equalization (CLAHE) further improves image contrast by redistributing pixel intensities, particularly useful for enhancing local contrast and delineating subtle bone structures. The processed images are then fused to integrate the benefits of different enhancements, balancing the effects of gamma correction and CLAHE to highlight relevant details while minimizing noise. Finally, noise reduction techniques are applied to further refine the image quality, ensuring clarity essential for accurate subsequent analysis.

To identify and isolate phalangeal regions critical for bone age assessment, a deep learning model pre-trained on Roboflow is employed for object detection within X-ray images. The model predicts regions of interest with a confidence threshold, extracting detections corresponding to phalangeal bones based on predefined class IDs. These detections are filtered to include only the relevant anatomical features, such as bones specific to hand and wrist development. The image is then annotated with these detections, marking the identified regions for subsequent extraction. Using these annotations, the image is masked to isolate the areas of interest, and contours are identified around these regions to precisely define their boundaries. A bounding box is computed to encapsulate all identified contours, facilitating the extraction of a focused region of interest (PROI) crucial for bone age assessment.

Resizing images to a consistent size is essential for uniformity in model training and prediction. The resizing process maintains the image's aspect ratio, ensuring that no distortion occurs during the transformation. Based on the target size defined, the function calculates the appropriate dimensions to either scale up or down the image proportionally. This calculation adjusts the image's height and width while preserving its original aspect ratio, thereby preventing any stretching or skewing that could compromise the integrity of anatomical structures. Additionally, the function pads the resized image to achieve the exact target dimensions if necessary, using a constant border value to maintain uniformity across all processed images. This standardized resizing process prepares the images for input into the bone age prediction model, ensuring consistent and reliable analysis across diverse datasets.

These processes collectively enable the transformation of raw X-ray images into standardized, enhanced representations suitable for accurate bone age prediction. By optimizing image clarity, isolating relevant anatomical regions, and maintaining uniformity in size, these preprocessing steps lay a robust foundation for the development and deployment of advanced machine learning models in medical diagnostics and research.

Figure 53 presents the code for Image Preprocessing and Dataset Preparation, which includes several essential functions. The first function applies gamma correction, adaptive histogram equalization, image fusion, and noise reduction to enhance the image quality and contrast. This preprocessing ensures that important features are highlighted while minimizing noise.

Additionally, the code features a function that extracts phalangeal regions of interest from images using the Roboflow API, allowing for precise isolation of key anatomical areas. Finally, the images are resized to a fixed aspect ratio of 256x256, ensuring consistency in input dimensions for the model. Together, these steps effectively prepare the dataset for training and evaluation, enhancing the model's ability to learn from the data.

Figure 54 demonstrates the results of image preprocessing, displaying the progression from the original image to the final output of resized phalangeal ROI extracted images.

```python
# Function to preprocess an image
def preprocess_image(image):
    # Convert the image to grayscale
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Gamma Correction
    gamma = 1.5
    inv_gamma = 1.0 / gamma
    table = np.array([((i / 255.0) ** inv_gamma) * 255 for i in np.arange(0, 256)]).astype("uint8")
    gamma_corrected_image = cv2.LUT(image, table)

    # Adaptive Histogram Equalization
    clahe = cv2.createCLAHE(clipLimit=10.0, tileGridSize=(8,8))
    clahe_image = clahe.apply(image)

    # Image Fusion
    fused_image = cv2.addWeighted(image, 0.5, gamma_corrected_image, 0.25, 0)
    fused_image = cv2.addWeighted(fused_image, 0.75, clahe_image, 0.25, 0)

    # Noise Reduction
    denoised_image = cv2.fastNlMeansDenoising(fused_image, None, 3, 7, 21)

    return denoised_image

# Function to extract phalangeal regions of interest (PROI) from an image
def extract_proi(image, img_path):
    # Perform inference on the image using the Roboflow model
    result = roboflow_model.predict(img_path, confidence=40).json()

    # Extract the detections from the result
    detections = sv.Detections.from_inference(result)

    # Filter the detections to only include the phalangeal regions of interest
    detections = detections[(detections.class_id == 0) | (detections.class_id == 6) | (detections.class_id == 7)]

    # Annotate the image with the detections
    mask_annotator = sv.MaskAnnotator()

    image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)

    # Annotate a mask with the detections
    mask = np.zeros_like(image)
    annotated_image = mask_annotator.annotate(scene=mask, detections=detections)

    # Extract the masked area from the image
    masked_image = np.where(annotated_image > 0, image, 0)

    # Convert to grayscale for finding contours
    gray = cv2.cvtColor(masked_image, cv2.COLOR_BGR2GRAY)

    # Find contours in the image
    contours, _ = cv2.findContours(gray, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Compute the bounding box of all the contours
    if len(contours) > 0:
        # Combine all the contours into one
        combined_contour = np.concatenate(contours)

        # Calculate the bounding box
        x, y, w, h = cv2.boundingRect(combined_contour)

        # Crop the image to the bounding box
        masked_image = masked_image[y:y+h, x:x+w]

    return masked_image

# Function to resize the image with a fixed aspect ratio
def resize_image(image, target_size=(256, 256)):
    # Resize the image with a fixed aspect ratio
    h, w = image.shape[:2]
    if h > w:
        new_h = target_size[0]
        new_w = int(w * new_h / h)
    else:
        new_w = target_size[1]
        new_h = int(h * new_w / w)

    resized_image = cv2.resize(image, (new_w, new_h))

    # Pad the image to the target size
    pad_h = (target_size[0] - new_h) // 2
    pad_w = (target_size[1] - new_w) // 2
    padded_image = cv2.copyMakeBorder(resized_image, pad_h, pad_h, pad_w, pad_w, cv2.BORDER_CONSTANT, value=[0, 0, 0])

    if padded_image.shape[:2] != target_size:
        padded_image = cv2.resize(padded_image, target_size)

    return padded_image
```

Figure 53: Code for Image Preprocessing and Dataset Preparation

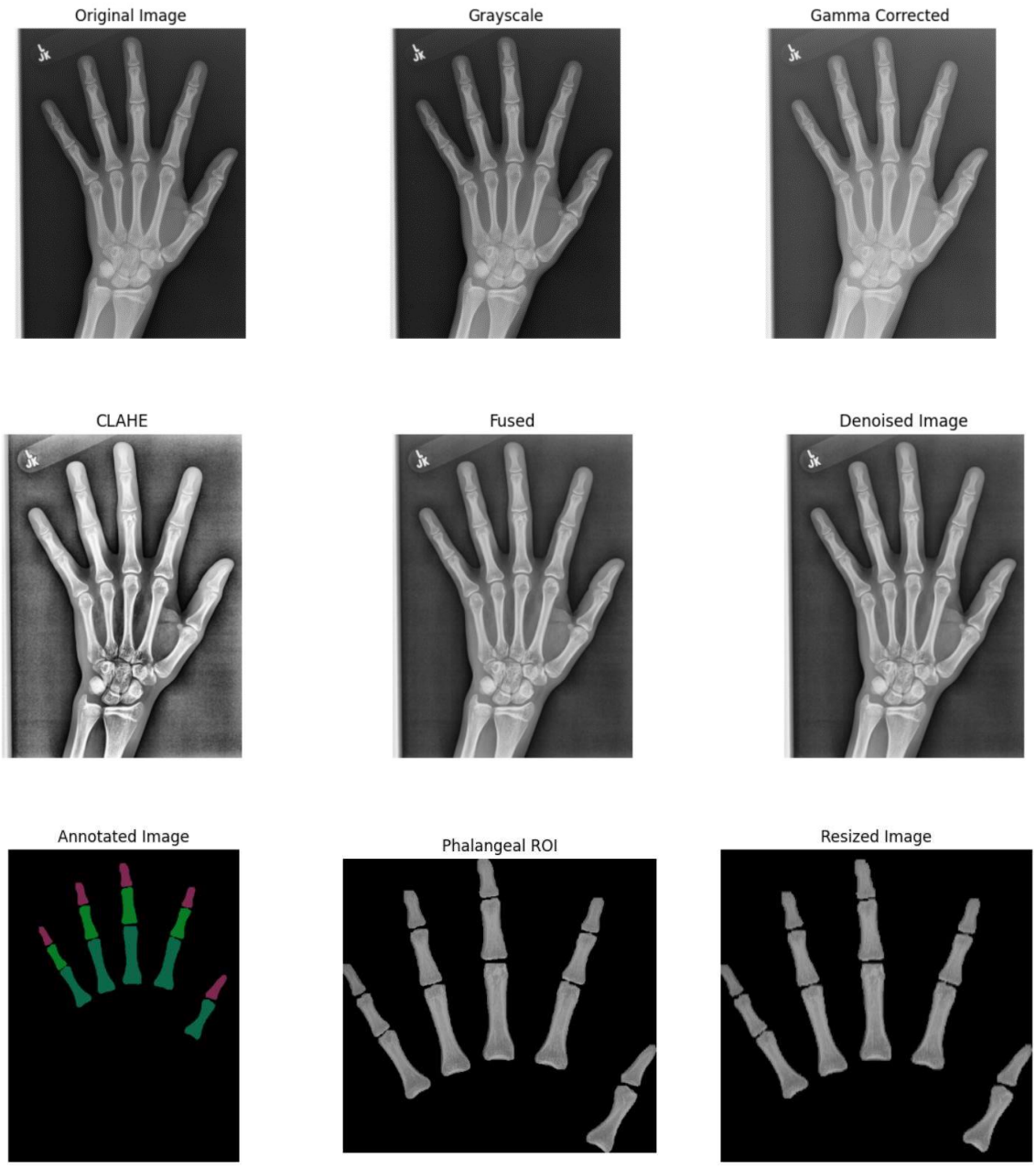


Figure 54: Results of Image Preprocessing

6.3.3 Model Building and Training

Build and Compile the Model

The model building process begins by loading the VGG16 base model, a widely recognized convolutional neural network pre-trained on the ImageNet dataset. This pre-trained model serves as a foundation for extracting relevant features from the X-ray images. The output from the last layer of the VGG16 model is then used as the input for a series of custom layers tailored to the task of bone age prediction.

As shown in the code snippet figure 55, the first step in preparing the dataset for training involves normalizing the pixel values, followed by splitting the dataset into training and validation sets.

Then, the model is built by adding custom layers on top of the VGG16 base model to fine-tune its capabilities. The first custom layer applied is the GlobalAveragePooling2D layer, which reduces the spatial dimensions (height and width) of the feature maps to a single vector, summarizing the presence of features detected by the convolutional layers. This is followed by a Dropout layer with a rate of 0.5, which helps prevent overfitting by randomly setting 50% of the input units to zero during each training update. Next, a Dense layer with 256 neurons and the ReLU activation function is added to introduce non-linearity and enable the model to learn complex patterns in the data. Additional Dense and Dropout layers are stacked to further enhance the model's capacity to capture intricate relationships within the data.

The model is created by combining the base VGG16 model with the newly added custom layers into a single, unified model as illustrated in code snippet figure 56. To preserve the valuable pre-trained features of the base model, its layers are frozen, preventing their weights from being updated during the training of the custom layers. The model is then compiled using the Adam optimizer, which adjusts the learning rate dynamically for efficient training, and the Mean Squared Error (MSE) loss function, suitable for the regression task of predicting continuous bone age values.

```python
# Prepare the training dataset
x_train = []
y_train = []

# Iterate over the training dataset
for index, row in train_labels.iterrows():
    img_name = str(row['id']) + '.png'
    img_path = os.path.join(train_image_path, img_name)
    image = cv2.imread(img_path)
    print(img_path)

    preprocessed_image = preprocess_image(image)
    proi = extract_proi(preprocessed_image, img_path)
    proi = resize_image(proi)

    # Normalize the pixel values to the range [0, 1]
    proi = proi / 255.0

    # Append the features to the training dataset
    x_train.append(proi)
    # Append the bone age to the target dataset
    y_train.append(row['boneage'])

# Convert to NumPy arrays
x_train = np.array(x_train)
y_train = np.array(y_train)

# Split the training dataset into training and validation datasets
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)
```

Figure 55: Code for Dataset Preparing

```python
# Funtion to build the model
def build_model():
    # Load the base model
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=(256, 256, 3))

    # Add custom layers on top of the base model
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.5)(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)

    # Output layer
    predictions = Dense(1, activation='linear')(x)

    # Create the model
    model = Model(inputs=base_model.input, outputs=predictions)

    # Freeze the base model
    for layer in base_model.layers:
        layer.trainable = False

    # Compile the model
    model.compile(optimizer=Adam(0.0001), loss='mean_squared_error', metrics=['mean_absolute_error'])

    return model
```

Python

Figure 56: Code for Building and Compiling Model

Training the Model

The training phase involves feeding the preprocessed and resized images, along with their corresponding bone age labels, into the compiled model. The dataset is split into training and validation sets to monitor the model's performance and adjust its parameters accordingly. Early stopping is implemented as a callback to monitor the validation loss, halting training if the loss does not improve for a specified number of epochs (patience) and restoring the best model weights encountered during training. Additionally, a ModelCheckpoint callback is used to save the model weights at each epoch if the validation loss improves, ensuring that the best-performing model is retained.

The model is trained over a predefined number of epochs, with batch processing employed to manage computational resources efficiently. During training, the model’s performance is evaluated based on the loss and Mean Absolute Error (MAE) metrics, providing insights into its accuracy and precision in predicting bone age. The code snippet for model training is presented in Figure 57.

```python
# Build the model
model = build_model()

# Define the callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
model_checkpoint = ModelCheckpoint('bone_age_model.weights.h5', monitor='val_loss', save_best_only=True, save_w


# Train the model
history = model.fit(
    x_train, y_train,
    validation_data=(x_val, y_val),
    epochs=25,
    batch_size=32,
    callbacks=[early_stopping, model_checkpoint]
    )

# Save the model
model.save(os.path.join(data_path, 'bone_age_model.keras'))
```

Python

Figure 57: Code for training model

6.3.5 Prediction

In this section, the process for loading the trained model, preparing the test dataset, and predicting the bone age is outlined.

First, the pre-trained model is loaded from the specified path. The test dataset is then prepared by iterating through each image in the test dataset. For each image, the image is read and preprocessed using the `preprocess_image` function. The phalangeal regions of interest (PROI) are extracted using the `extract_proi` function, and the image is resized to the required dimensions using the `resize_image` function. The pixel values of the processed images are normalized to the range [0, 1] before being appended to the test dataset.

After preprocessing all test images, the test dataset is converted to a NumPy array. The model is then used to predict the bone age for each image in the test dataset. The predicted bone ages are rounded to the nearest integer for easier interpretation.

Finally, the predictions are saved to a CSV file. The predicted bone ages are added to the test labels DataFrame, and the updated DataFrame is saved to a file named 'predictions.csv'. This file contains the Case ID along with the corresponding predicted bone age, facilitating further analysis and reporting. Figure 58 displays the code snippet for prediction process.

```python
# Load the model
model_path = os.path.join(data_path, 'bone_age_model.keras')
model = load_model(model_path)

# Prepare the test dataset
x_test = []

# Iterate over the test dataset
for index, row in test_labels.iterrows():
    img_name = str(row['Case ID']) + '.png'
    img_path = os.path.join(test_image_path, img_name)
    image = cv2.imread(img_path)
    print(img_path)

    preprocessed_image = preprocess_image(image)
    proi = extract_proi(preprocessed_image, img_path)
    proi = resize_image(proi)

    # Normalize the pixel values to the range [0, 1]
    proi = proi / 255.0

    # Append the features to the test dataset
    x_test.append(proi)

# Convert to NumPy arrays
x_test = np.array(x_test)

# Predict the bone age
y_test = model.predict(x_test)
y_test_rounded = np.round(y_test).astype(int)

# Save the predictions to a CSV file
test_labels['Predicted Bone Age'] = y_test_rounded
test_predictions_path = os.path.join('predictions.csv')
test_labels.to_csv(test_predictions_path, index=False)
```

Python

Figure 58: Code for Prediction

6.3.6 Attention Map Visualization

The final phase involves prediction and visualization, specifically using Grad-CAM to generate heatmaps that help understand model decisions. The process begins with the model being loaded from a specified path. For each image in the test dataset, the phalangeal region of interest (PROI) is extracted and preprocessed using the same functions applied during training.

Once the images are preprocessed, normalized, and resized, they are fed into the model to predict bone age. Grad-CAM is then utilized to generate a heatmap that highlights the regions of the image that are most influential in the model's prediction. This involves computing the gradients of the top predicted class and creating a class activation map (CAM) that is overlaid on the original image to visualize the areas the model focused on.

The generated heatmaps are then overlaid on the original PROI images to provide a visual representation of the areas that contributed most to the prediction. The code snippet for attention map visualization is presented in Figure 5.

```
# Function to compute the Grad-CAM heatmap
def compute_gradcam(model, img_array, layer_name):
    grad_model = Model(model.inputs, [model.get_layer(layer_name).output, model.output])

    # Compute the gradient of the top predicted class for the input image
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(img_array)
        loss = predictions

    # Extract the output and gradients
    output = conv_outputs[0]
    grads = tape.gradient(loss, conv_outputs)[0]
    gate_f = tf.cast(output > 0, 'float32')
    gate_r = tf.cast(grads > 0, 'float32')
    guided_grads = gate_f * gate_r * grads

    # Average the gradients spatially
    weights = tf.reduce_mean(guided_grads, axis=(0, 1))
    cam = np.zeros(output.shape[:2], dtype=np.float32)

    # Multiply the output feature map with the computed weights
    for i, w in enumerate(weights):
        cam += w * output[:, :, i]

    cam = cv2.resize(cam.numpy(), (img_array.shape[2], img_array.shape[1]))
    cam = np.maximum(cam, 0)

    heatmap = (cam - cam.min()) / (cam.max() - cam.min())
    heatmap = np.uint8(255 * heatmap)


    # Apply a colormap to the heatmap
    heatmap = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)

    return heatmap

# Function to overlay the heatmap on the image
def overlay_heatmap(image, heatmap, alpha=0.4, colormap=cv2.COLORMAP_JET):
    heatmap = cv2.applyColorMap(heatmap, colormap)
    heatmap = heatmap.astype(np.float32) / 255.0
    overlayed_image = cv2.addWeighted(image, 1-alpha, heatmap, alpha, 0)
    return overlayed_image

# Load the model
model_path = os.path.join(data_path, 'bone_age_model.keras')
model = load_model(model_path)

for index, row in test_labels.iterrows():
    img_name = str(row['Case ID']) + '.png'
    img_path = os.path.join(test_image_path, img_name)
    image = cv2.imread(img_path)
    print(img_path)

    # Process the image and extract the phalangeal region of interest (PROI)
    preprocessed_image = preprocess_image(image)
    proi = extract_proi(preprocessed_image, img_path)
    proi = resize_image(proi)

    # Normalize the pixel values to the range [0, 1]
    proi = proi.astype(np.float32) / 255.0

    # Reshape for model input
    img_array = np.expand_dims(proi, axis=0)

    # Compute Grad-CAM
    heatmap = compute_gradcam(model, img_array, 'block5_conv3')

    # Overlay heatmap on the image
    overlayed_image = overlay_heatmap(proi, heatmap)
    overlayed_image = np.where(proi > 0, overlayed_image, 0)
  # Plot the PROI and Grad-CAM heatmap
  fig, ax = plt.subplots(1, 2, figsize=(10, 5))
  ax[0].imshow(proi, cmap='gray')
  ax[0].set_title('Phalangeal Region of Interest')
  ax[0].axis('off')
  ax[1].imshow(cv2.cvtColor(overlayed_image, cv2.COLOR_BGR2RGB))
  ax[1].set_title('Grad-CAM Heatmap')
  ax[1].axis('off')
  plt.savefig(f'gradcam_{img_name}')
  plt.show()
```

```python
# Function to overlay the heatmap on the image
def overlay_heatmap(image, heatmap, alpha=0.4, colormap=cv2.COLORMAP_JET):
    heatmap = cv2.applyColorMap(heatmap, colormap)
    heatmap = heatmap.astype(np.float32) / 255.0
    overlayed_image = cv2.addWeighted(image, 1-alpha, heatmap, alpha, 0)
    return overlayed_image

# Load the model
model_path = os.path.join(data_path, 'bone_age_model.keras')
model = load_model(model_path)

for index, row in test_labels.iterrows():
    img_name = str(row['Case ID']) + '.png'
    img_path = os.path.join(test_image_path, img_name)
    image = cv2.imread(img_path)
    print(img_path)

    # Process the image and extract the phalangeal region of interest (PROI)
    preprocessed_image = preprocess_image(image)
    proi = extract_proi(preprocessed_image, img_path)
    proi = resize_image(proi)

    # Normalize the pixel values to the range [0, 1]
    proi = proi.astype(np.float32) / 255.0

    # Reshape for model input
    img_array = np.expand_dims(proi, axis=0)

    # Compute Grad-CAM
    heatmap = compute_gradcam(model, img_array, 'block5_conv3')

    # Overlay heatmap on the image
    overlayed_image = overlay_heatmap(proi, heatmap)
    overlayed_image = np.where(proi > 0, overlayed_image, 0)
```

Figure 59: Code for Attention Map Generation

Figure 60 shows the results of the generated attention map using Grad-CAM. The colors in the heatmaps represent the relative importance of different regions of the image in the model's prediction:

- **Red**: Areas with high red intensity contribute significantly to the prediction.
- **Blue**: Areas with high blue intensity contribute less to the prediction.
- **Intermediate**: These represent varying levels of contribution.

In the provided images, the red and orange regions in the phalanx area indicate that these specific areas are most critical for the model in determining bone age.

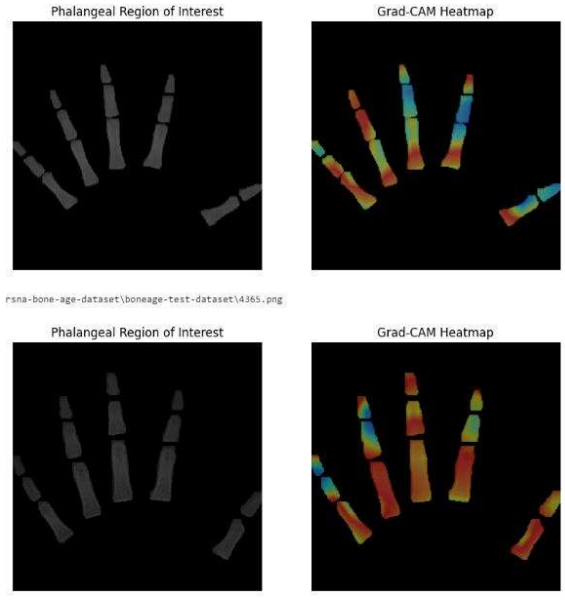


Figure 60: Results of Attention Map

6.4 Adult height prediction model

This module focuses on predicting adult height using the Bayley-Pinneau(BP) method and enhances its accuracy through Machine Learning techniques, with a further development towards a neural network-based approach. This chapter outlines the implementation steps including data preprocessing, BP method application, data distribution analysis, ML model development and neural network enhancement.

6.4.1 Data Preprocessing

Data Collection

For this project, longitudinal data from the Ogi study of healthy children born around 1975 in Northern Japan was utilized. This dataset contains information relevant to height prediction, such as actual age, bone age, current height, and adult height.

To integrate the Bayley-Pinneau method for adult height prediction within the system, several BP tables were loaded. These tables, specific to gender and categorized by bone age delay (the difference between actual age and bone age), provide reference points for predicting adult height. Three separate tables were loaded for each gender: 'average', 'accelerated', and 'retarded', reflecting different bone age development trajectories. Figure 61 shows the code snippets for loading the dataset and BP tables.

Figure 61: Load datasets

Descriptive statistics were calculated as displayed in figure 62 to understand the distribution and summary statistics of the dataset.

	Bone Age(y)	Chronological Age(y)	Current Height(cm)	Adult Height(cm)
count	3865.000000	3865.000000	3865.000000	3865.000000
mean	12.364812	12.325433	145.287245	168.768875
std	3.638514	3.739350	17.498739	9.601923
min	6.000000	5.000000	100.000000	139.000000
25%	9.000000	8.900000	130.000000	162.000000
50%	12.500000	12.600000	149.000000	169.000000
75%	15.600000	15.500000	159.000000	176.000000
max	18.700000	20.000000	181.000000	192.000000

Figure 62: Describe the dataset

Loaded data is visualized as below figure 63 including sex, bone age, chronological age, current height and adult height. Additionally figure 64 shows the basic information about the dataset such as the columns, data row and other attributes related to data.

	Sex	Bone Age(y)	Chronological Age(y)	Current Height(cm)	Adult Height(cm)
0	FEMALE	6.0	7.0	120.0	164.0
1	FEMALE	6.0	7.3	118.0	160.0
2	FEMALE	6.0	6.3	106.0	149.0
3	FEMALE	6.0	7.0	110.0	149.0
4	FEMALE	6.0	6.2	119.0	169.0
...	...	...	...	...	...
3860	MALE	18.7	18.7	170.0	179.0
3861	MALE	18.7	18.4	161.0	179.0
3862	MALE	18.7	17.5	164.0	179.0
3863	MALE	18.7	17.9	165.0	181.0
3864	MALE	18.7	18.0	157.0	172.0

3865 rows x 5 columns

Figure 63: Visualize the dataset

```
[12] # Display basic information about the dataset
     print(df.info())

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 610 entries, 0 to 609
     Data columns (total 5 columns):
      #   Column                Non-Null Count  Dtype
     ---  ------                --------------  -----
      0   Sex                   610 non-null    object
      1   Bone Age(y)           610 non-null    float64
      2   Chronological Age(y)  610 non-null    float64
      3   Current Height(cm)    610 non-null    float64
      4   Adult Height(cm)      610 non-null    float64
     dtypes: float64(4), object(1)
     memory usage: 24.0+ KB
     None
```

Figure 64: Basic information about the dataset

Data Cleaning

Data cleaning involved handling missing values, ensuring the correct data types, and normalizing the data. Python's pandas library was used extensively for these operations.

Identify and handle missing values in the dataset. This can be done by inputting the missing values with mean, median, or mode, or by removing the rows or columns containing missing values.

```
# Check for missing data
missing_data = df.isnull().sum()

# Display the count of missing values for each column
print("Missing Data Count:")
print(missing_data)
```

```
Missing Data Count:
Sex                     0
Bone Age(y)             0
Chronological Age(y)    0
Current Height(cm)      0
Adult Height(cm)        0
dtype: int64
```

Figure 65: Check for missing values

As the figure 65 shows, the dataset didn't contain any missing values and there was no need for data inputting.

Data Visualization

Various visualizations were created to gain insights into the dataset, including histograms, boxplots, and scatter plots.

Plotting histograms for each feature is a crucial step in data exploration and visualization. It helps to determine if the data is skewed to the left or right, which can affect the analysis and then to identify potential outliers in the data. Plotting histograms for each feature as illustrated in figure 66 gave valuable insights into the data and how to proceed with the analysis.

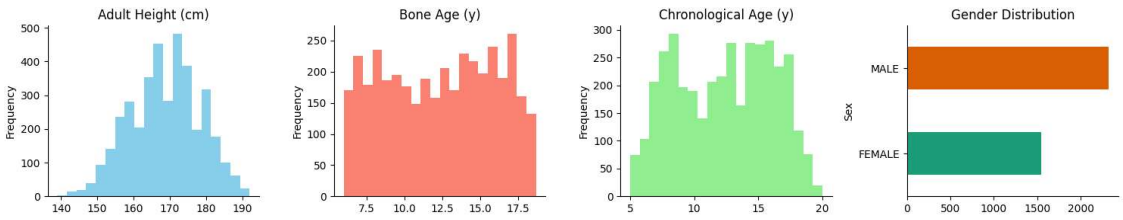


Figure 66: Plot the histograms for each attribute and target values

The distribution of bone age (BA) plots provides an overview of the skeletal maturity levels across the dataset. These plots typically show how bone age is spread across different age groups. In this context, the distribution helps identify any skewness or anomalies in the data, such as clusters of subjects with significantly advanced or delayed bone ages compared to their chronological age. The results from these plots indicate that most subjects fall within the

expected range of bone ages for their respective chronological ages, with a few outliers showing advanced or delayed skeletal maturity.

The distribution of adult height plots as displayed in figure 67, illustrates how the final adult heights are spread across the population in the dataset. These plots often reveal the central tendency (mean or median), variability (standard deviation), and any potential outliers in the adult height data. The results from the distribution plots in the dataset show a normal distribution of adult heights, indicating a well-balanced sample with few extreme outliers. This distribution is crucial for validating that the dataset is representative of the population being studied

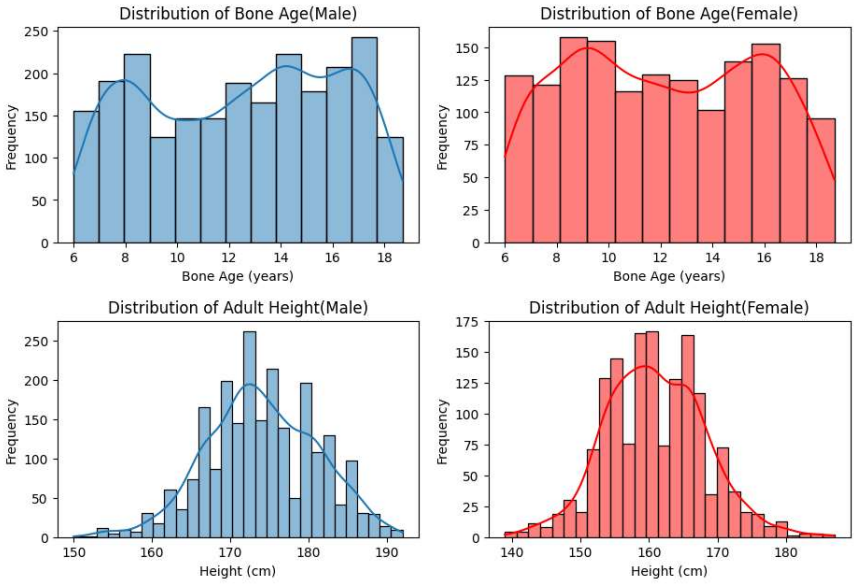


Figure 67: Plot boxplots for each attribute

Pair plots offer a visual representation of the relationships between multiple variables by plotting them against each other in a grid of scatterplots. The generated pai plots are displayed in figure 68. In the context of adult height prediction, pair plots may include variables like bone age, chronological age, current height, and predicted adult height. These plots help in identifying correlations, linear or nonlinear relationships, and any patterns that might suggest interactions between the variables. The results from the pair plots reveal strong linear relationships between bone age and chronological age, and between current height and adult

height. These relationships confirm the logical connections between these variables, supporting their inclusion in the predictive models.

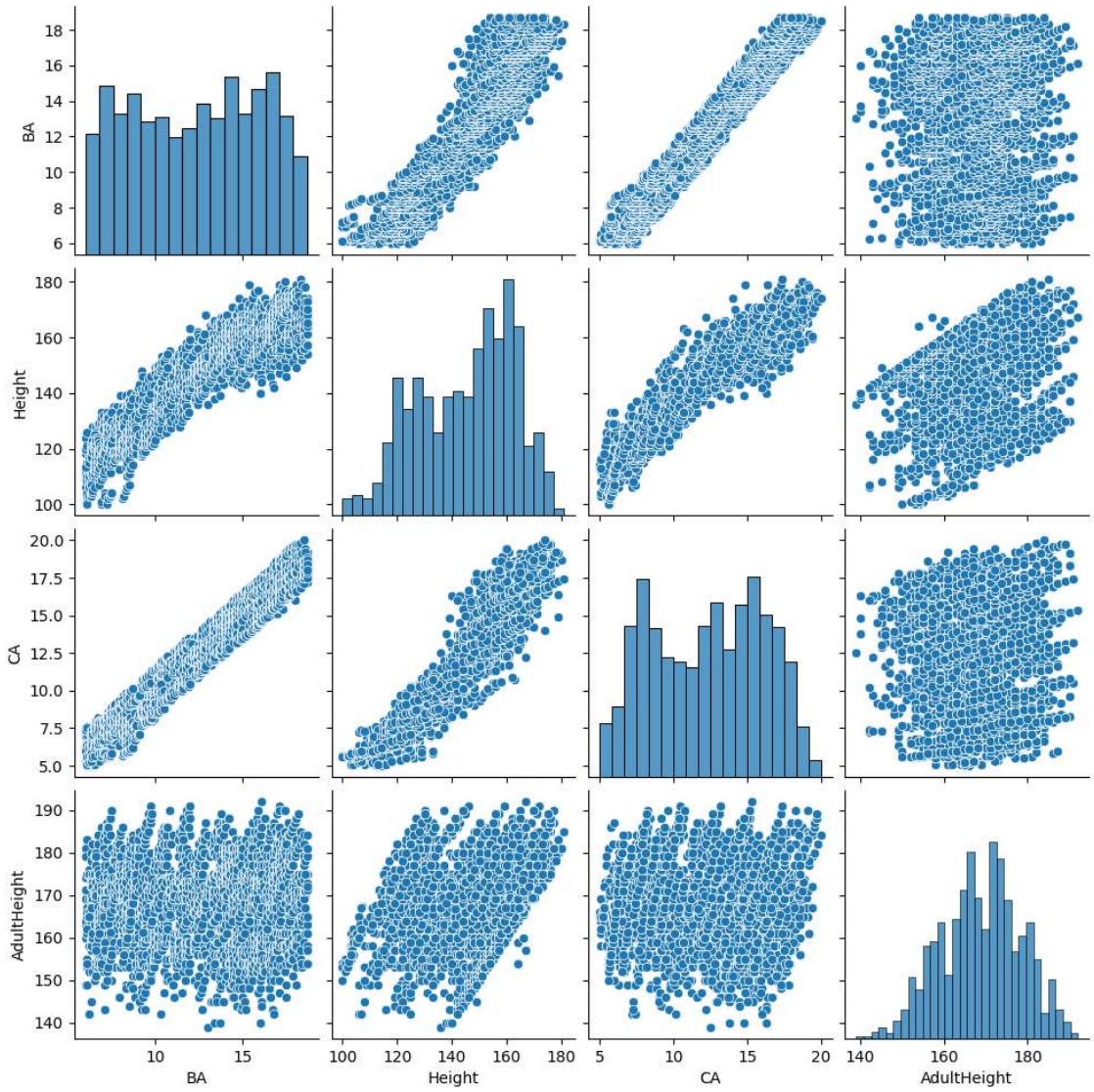


Figure 68: Pair plots

The correlation matrix is a visual representation of the correlation coefficients between pairs of variables in the dataset. It helps in understanding the strength and direction of linear relationships between the variables, with values ranging from -1 to 1. A correlation close to 1 indicates a strong positive relationship, while a correlation close to -1 indicates a strong negative relationship. The results from the correlation matrix show high positive correlations between current height and adult height, and between bone age and chronological age as illustrated in figure 69. These strong correlations suggest that these variables are significant predictors of adult height and should be given priority in the modeling process. Additionally,

the matrix may reveal any multicollinearity issues, where two or more predictors are highly correlated, potentially necessitating further data processing or model adjustments.

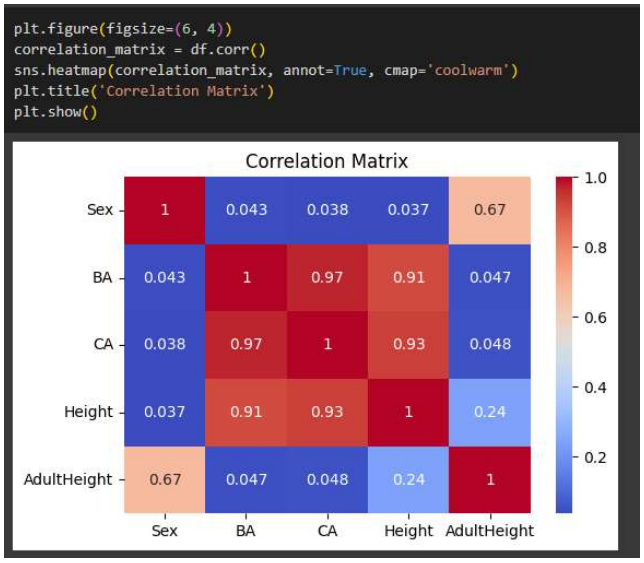


Figure 69: Heat map for attributes

The heatmap showcases a strong positive correlation between 'Height' and 'AdultHeight,' as evidenced by the deep red hue approaching 1.0. This suggests that individuals with greater initial height are more likely to attain greater adult height. A weak positive correlation, indicated by the light-yellow shade near 0.2, is observed between 'CA' and 'Height.' While the interpretation of this association is limited without further context, it hints at a possible slight positive trend where higher values in 'CA' may correlate with marginally greater heights. The remaining correlations depicted in the heatmap appear insignificant, with colors close to white signifying weak or nonexistent relationships.

6.4.2 Adult height prediction using Bayley-Pineau Method

The data underwent several preprocessing steps to prepare it for BP prediction analysis. First, columns were renamed for clarity and ease of use. The data was then divided into two subsets based on gender, with '0' representing females and '1' representing males.

For each gender-specific dataset, a new 'BA_delay' column was calculated by subtracting the 'BA' value from the 'CA' value. This represents the delay between the chronological age and the actual growth. Finally, the 'BA_delay' was categorized into three groups: 'Normal' (delay less than 1), 'Delayed' (delay greater than 1), and 'Advanced' (delay between -1 and 1). A function was applied row-wise to achieve this categorization. The resulting distribution of these

categories was then analyzed by calculating the value counts for the 'BA_category' column. The code snippet for BA calculation and dividing data is displayed in figure 70.

```
# Filter for females
data_female = data_female[data_female['Sex'] == 0]

# Calculate BA delay
data_female.loc[:, 'BA_delay'] = data_female['CA'] - data_female['BA']

# Categorize BA delay
def categorize_ba_delay(row):
    if abs(row['BA_delay']) < 1:
        return 'Normal'
    elif row['BA_delay'] > 1:
        return 'Delayed'
    else:
        return 'Advanced'

data_female.loc[:, 'BA_category'] = data_female.apply(categorize_ba_delay, axis=1)

# Get count of each category
category_counts = data_female['BA_category'].value_counts()
print(category_counts)
```

Figure 70: BA delay calculation and divide data

BP method utilizes a specific notation for bone age that combines years and months, and the notation used in the data set is different. A function was created for transforming the skeletal age values from the dataset into a format compatible with the Bayley-Pinneau (BP) method as displayed in figure 71.

```
def convert_skeletal_age_to_bp_format(skeletal_age):
    years = int(skeletal_age)
    months_decimal = (skeletal_age - years) * 12

    if years >= 17:
        if months_decimal >= 0 and months_decimal < 6:
            months = "0"
        else:
            months = "6"
        if years >= 18:
            return "18-0"
    elif years >= 9:
        if months_decimal >= 0 and months_decimal < 3:
            months = "0"
        elif months_decimal >= 3 and months_decimal < 6:
            months = "3"
        elif months_decimal >= 6 and months_decimal < 9:
            months = "6"
```

```
        else:
            months = "9"
    else:
        if months_decimal >= 0 and months_decimal < 3:
            months = "0"
        elif months_decimal >= 3 and months_decimal < 6:
            months = "3"
        elif months_decimal >= 6 and months_decimal < 10:
            months = "6"
        else:
            months = "10"

    return f"{years}-{months}"
```

Figure 71: Code to convert skeletal age to BP format

Then another function was implemented to first convert the skeletal age from the dataset into the format expected by the BP tables. Then, it retrieves the minimum and maximum skeletal ages represented in the table.

The function checks if the provided skeletal age falls outside the table's range. If it's lower than the minimum, the PMH (percentage maturity height) corresponding to the minimum skeletal age is used. Conversely, for ages exceeding the maximum, a PMH of 100% is assumed. Otherwise, the function attempts to look up the PMH value directly from the table using the converted skeletal age string as the key. If the exact skeletal age isn't present in the table, a default value (PMH of the minimum age) is returned.

The adult height prediction function utilizes the PMH function to retrieve the PMH for each data point. It iterates through each row in the data, extracting the skeletal age, current height (converted to inches), and BA category. Depending on the BA category ('Normal', 'Advanced', or 'Delayed'), the appropriate BP table (normal, advanced, or delayed) is used to obtain the PMH via PMH function.

The predicted adult height is then calculated using the PMH. Finally, the predicted height is converted back to centimeters. If the PMH cannot be determined, a 'None' value is assigned as the predicted height for that data point. The function returns a list containing the predicted adult heights for all data points.". Figure 72 illustrates the functions to derive PMH and predict the adult height.

```
def get_pmh(bp_table, skeletal_age):
    # Convert skeletal age to BP table format
    skeletal_age_str = convert_skeletal_age_to_bp_format(skeletal_age)
    # Find the minimum and maximum skeletal ages in the BP table
    min_sa = float(bp_table.columns[1].split('-')[0]) + float(bp_table.columns[1].split('-')[1])/12
    max_sa = float(bp_table.columns[-1].split('-')[0]) + float(bp_table.columns[-1].split('-')[1])/12
    # Check if skeletal age is beyond the table range
    if skeletal_age < min_sa:
        pmh = bp_table.iloc[0, 1]
    elif skeletal_age > max_sa:
        pmh = 100.0
    else:
        if skeletal_age_str in bp_table.columns:
            pmh = bp_table[skeletal_age_str].iloc[0]
        else:
            pmh = None
    if pmh is None:
        pmh = bp_table.iloc[0, 1]
    try:
        return float(pmh)
    except ValueError:
        return None
def predict_adult_height(data, bp_table_normal, bp_table_advanced, bp_table_delayed):
    predicted_heights = []
    for _, row in data.iterrows():
        skeletal_age = row['BA']
        current_height = round(row['Height'] * 0.393701, 2)
        if row['BA_category'] == 'Normal':
            pmh = get_pmh(bp_table_normal, skeletal_age)
        elif row['BA_category'] == 'Advanced':
            pmh = get_pmh(bp_table_advanced, skeletal_age)
        else:
            pmh = get_pmh(bp_table_delayed, skeletal_age)
        print(f"Skeletal Age: {skeletal_age}, Current Height: {current_height}, PMH: {pmh}")
        if pmh:
            predicted_height = current_height / (pmh / 100)
            predicted_height_cm = round(predicted_height * 2.54, 2)
            predicted_heights.append(predicted_height_cm)
        else:
            predicted_heights.append(None)
    return predicted_heights
```

Figure 72: Code to get PMH and predict height

To ensure compatibility for calculations, data types in the female dataset (BA, Height, AdultHeight) and BP tables were converted to numeric using pd.to_numeric (handling errors with 'coerce'). Then, the predict_adult_height function assigned predicted adult heights to the 'PredictedHeight' column in the female data, considering BA category to select the appropriate BP table (average, accelerated, or retarded) from the pre-loaded ones as shown in figure 73.

```
# Ensure correct data types in the dataset
data_female.loc[:, 'BA'] = pd.to_numeric(data_female['BA'], errors='coerce')
data_female.loc[:, 'Height'] = pd.to_numeric(data_female['Height'], errors='coerce')
data_female.loc[:, 'AdultHeight'] = pd.to_numeric(data_female['AdultHeight'], errors='coerce')

# Ensure correct data types in BP tables
bpf_average = bpf_average.apply(pd.to_numeric, errors='coerce', axis=1)
bpf_accelerated = bpf_accelerated.apply(pd.to_numeric, errors='coerce', axis=1)
bpf_retarded = bpf_retarded.apply(pd.to_numeric, errors='coerce', axis=1)

# Apply BP tables
data_female.loc[:, 'PredictedHeight'] = predict_adult_height(data_female, bpf_average, bpf_accelerated, bpf_retarded)
print(data_female[['BA', 'Height', 'PredictedHeight']].head())
```

Figure 73: Code to apply BP tables

6.4.3 Analysis of the results from BP method

To assess the model's accuracy in predicting adult height, deviations between predicted and actual heights were calculated. It is calculated by the difference between predicted height and adult height as displayed in figure 74.

```
# Calculate deviation
data_female['Deviation'] = data_female['PredictedHeight'] - data_female['AdultHeight']
```

Figure 74: Calculate deviation

Summary statistics of the deviations were generated to get a high-level understanding of the prediction errors. These statistics, typically including measures like mean, standard deviation, minimum, and maximum values, provide insights into the distribution and spread of the errors.

```
# Summary statistics
summary_stats = data_female['Deviation'].describe()
print(summary_stats)
```

Figure 75: Summary statistics

The calculated summary statistics of the prediction deviations as illustrated in code snippet figure 75, provide insights into the model's performance. The average deviation is -0.42 cm, indicating a slight tendency for the model to under-predict adult height by a small margin. The standard deviation of 2.53 cm highlights some variability in the errors, with a range of -10.92 cm to 12.77 cm. The distribution is further characterized by the quartiles: the middle 50% of deviations fall between -1.50 cm and 0.37 cm, suggesting that a significant portion of the predictions are fairly close to the actual heights.

A histogram of the deviations was plotted to visualize the distribution of prediction errors as shown in figure 76.

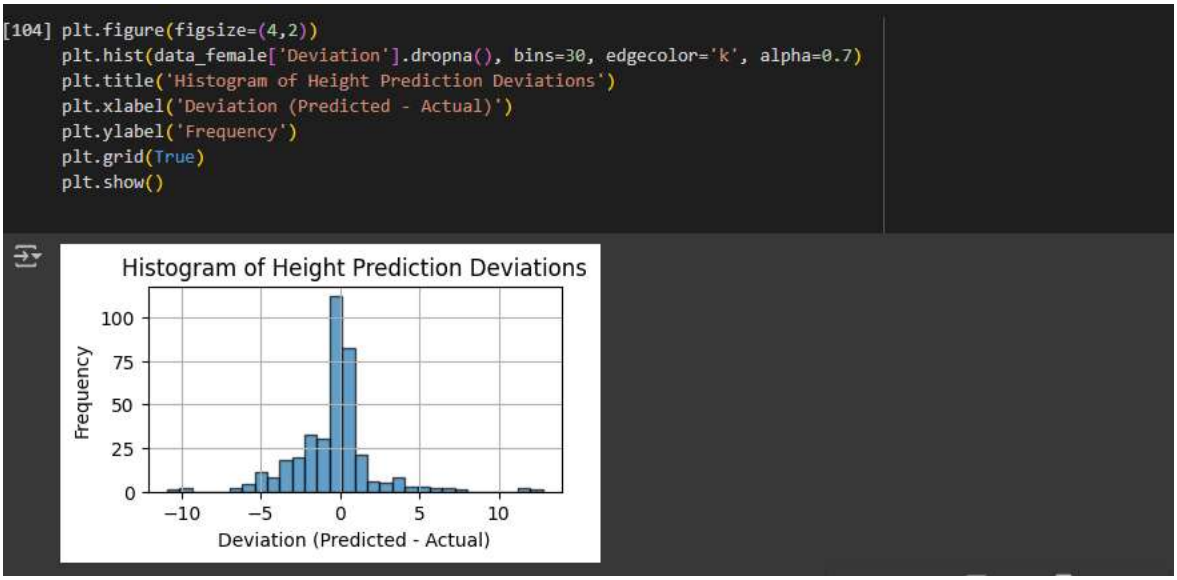


Figure 76: Histogram of height prediction deviation

The histogram shows the distribution of the prediction errors, with the x-axis representing the deviation (predicted minus actual height in centimeters) and the y-axis representing the frequency of each error value.

The distribution appears centered slightly to the left of zero, with a tail extending towards larger negative deviations. This indicates a tendency for the model to under-predict adult height in some cases. There are more frequent errors between -1.5 cm and 0 cm, and the number of errors reduces as the magnitude of the deviation increases. While there are some scattered errors throughout the range, the overall pattern suggests that the model makes reasonably accurate predictions for a substantial portion of the data, with a slight bias towards under-prediction.

The scatter plot displayed in figure 77 visualizes the correlation between predicted and actual adult heights for the female data points.

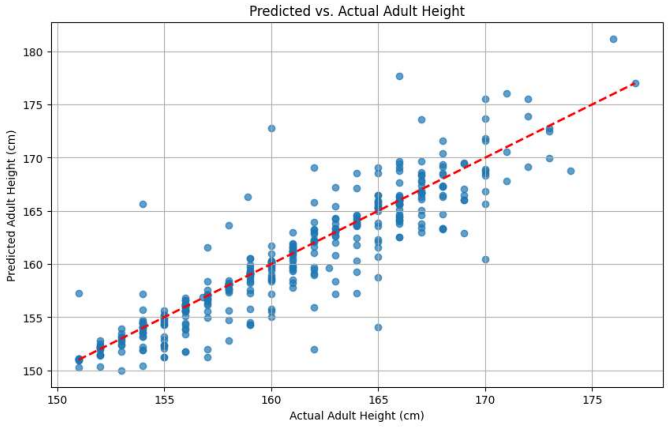


Figure 77: Scatter plot between predicted vs Actual adult Height

In this scatter plot, we can see that most of the data points cluster around the diagonal line, although there is some scattering. This indicates a generally positive correlation between predicted and actual heights, but also some deviations from the ideal perfect prediction. For data points that fall above the diagonal line, the model over-predicted the adult height. Conversely, data points below the line represent under-predictions. The spread of the data points around the diagonal line suggests that the model's accuracy varies somewhat, with some predictions being more precise than others

6.4.4 Enhancing prediction using Neural Network Model

While the BP method provides a traditional approach to height prediction, its accuracy is limited by the inherent assumptions and the coarse categorization of growth patterns. To overcome these limitations, a neural network model is developed to enhance the predictions by incorporating additional features and leveraging modern machine learning techniques.

Data Preparation

Data is prepared before applying machine learning models as illustrated in code snippet figure 78. This involves splitting the data into features (X) and target (y) variables, followed by dividing the data into training and testing sets.

```
data_female['gp'] = (df['AdultHeight'] - df['Height']) / df['AdultHeight']
data_male['gp'] = (df['AdultHeight'] - df['Height']) / df['AdultHeight']

# Function to split the data into train, validation, and test sets
def split_data(data):
    train_data, temp_data = train_test_split(data, test_size=0.3, random_state=42)
    val_data, test_data = train_test_split(temp_data, test_size=0.5, random_state=42)
    return train_data, val_data, test_data

# Split the dataset
train_female, val_female, test_female = split_data(data_female)
train_male, val_male, test_male = split_data(data_male)
```

Figure 78: Code for data preparation

Neutral Network Architecture

The neural network architecture developed for predicting adult height is carefully designed to capture the complex relationships between various growth-related features. This architecture is tailored specifically for male and female growth patterns, recognizing the inherent differences in their development trajectories. The input layer of the neural network consists of five neurons, each corresponding to a key feature: bone age (BA), bone age delay (BA_delay), growth potential (gp), percentile maturity height (PMH), and current height (H). These features are crucial for understanding the growth process and provide the necessary inputs for the model to make accurate predictions.

For the female model, the architecture includes three hidden layers with 128, 64, and 32 neurons, respectively. Each hidden layer is followed by a dropout layer, which is implemented to prevent overfitting by randomly disabling a fraction of neurons during training. The ReLU (Rectified Linear Unit) activation function is used in the hidden layers. ReLU is chosen because it introduces non-linearity into the model, allowing it to learn complex patterns from the data, which is essential for accurately predicting adult height.

The male model has a more complex structure, with four hidden layers consisting of 256, 128, 64, and 32 neurons. The additional layer and increased number of neurons reflect the need to capture the more complex growth patterns typical of males. Similar to the female model, dropout layers are used after each hidden layer, and ReLU is employed as the activation function to enhance the model's learning capacity.

The output layer of both models consists of a single neuron with a linear activation function, which outputs the predicted growth potential or adult height. This linear function is appropriate for regression tasks like height prediction, where the output is a continuous value. The model

is compiled using the Adam optimizer, known for its efficiency in adjusting the learning rate during training, leading to faster convergence and improved performance. The Mean Squared Error (MSE) is used as the loss function, which effectively penalizes larger errors, making it suitable for minimizing the difference between the predicted and actual heights.

Code implementation for female dataset is shown in figure 79 and the code implementation for male dataset is displayed in figure 80.

```
# Define features and target
features = ['BA', 'BA_delay', 'gp', 'PMH', 'Height']
target = 'AdultHeight'

# Build and train the model
modelf = Sequential([
    Dense(128, input_dim=len(features), activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(1)  # Output layer for adult height prediction
])
modelf.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Figure 79: Code for training model for female dataset

```
# Define features and target
features = ['BA', 'BA_delay', 'gp', 'PMH', 'Height']
target = 'AdultHeight'

# Build and train the model
modelm = Sequential([
    Dense(256, input_dim=len(features), activation='relu'),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(1)  # Output layer for adult height prediction
])
modelm.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Figure 80: Code for training model for male dataset

Training and Validation

The training and validation of the neural network models involve several critical steps designed to optimize the models' performance while preventing overfitting. The process begins with a train-test split of the dataset, where the data is divided into training and validation sets. This

split ensures that the model is trained on one subset of the data and validated on another, allowing for an unbiased evaluation of its performance.

Early stopping is implemented during training to prevent overfitting. Early stopping monitors the model's performance on the validation set, halting the training process when improvements plateau. This technique is essential for ensuring that the model does not become too tailored to the training data, which would reduce its ability to generalize to new, unseen data.

The batch size and number of epochs are also carefully chosen to balance training efficiency and model performance. For the female model, a batch size of 32 is used, while the male model is trained with a smaller batch size of 16. This difference is due to the more complex architecture of the male model, which benefits from smaller batch sizes in learning the intricate patterns in the data. Both models are trained for up to 250 epochs, although the early stopping mechanism typically halts training before reaching this limit, once the validation loss stops improving.

Performance evaluation is conducted after training, using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared score. These metrics provide a quantitative assessment of the model's accuracy and reliability.

The MAE indicates the average error in predicting adult height, while the MSE highlights the extent of larger errors, and the R-squared score measures how well the model explains the variance in the data.

Code implementation for male model is illustrated in figure 81. whereas the implementation for female model is shown in figure 82.

```
history = modelm.fit(
    train_male[features], train_male[target],
    validation_data=(val_male[features], val_male[target]),
    epochs=250, batch_size=16,
    callbacks=[EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=True)]
)

test_loss, test_mae = modelm.evaluate(test_male[features], test_male[target])
print(f'Male Model Test MAE: {test_mae}')
```

Figure 81: Code for male model

```
history = modelf.fit(
    train_female[features], train_female[target],
    validation_data=(val_female[features], val_female[target]),
    epochs=250, batch_size=32,
    callbacks=[EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=True)]
)

test_loss, test_mae = modelf.evaluate(test_female[features], test_female[target])
print(f'Female Model Test MAE: {test_mae}')
```

Figure 82: Code for female model

By following this structured approach, the neural network models are not only trained effectively but also validated thoroughly, ensuring their readiness for deployment in predicting adult height with high accuracy.

6.5 Summary

This chapter details the development of our research project, outlining the steps taken and results obtained for each module:1. carpal bone age prediction, 2. phalangeal bone age prediction, and 3. adult height prediction. We explored various machine learning algorithms, image processing techniques and neural networks in building these modules.

Chapter 7

Evaluation

7.1 Introduction

This chapter will describe the experiments that were conducted in each module. The results obtained in each experiment are explained with the level of accuracy of the module.

7.2 Carpal bone Age Prediction Module

Evaluation of the Carpal Bone Age Prediction Model was carried out using a test dataset consisting of hand X-ray images. The evaluation process involved comparing the model's predicted bone ages with the actual bone ages provided in the test dataset.

To evaluate the model's performance, several metrics were calculated:

- **Mean Absolute Error (MAE):** The average of the absolute differences between predicted and actual bone ages, providing a straightforward measure of prediction accuracy.
- **Mean Squared Error (MSE):** The average of the squared differences between predicted and actual values, reflecting the variance in the prediction errors.
- **Root Mean Squared Error (RMSE):** The square root of the MSE, offering an interpretable measure of the typical error magnitude.
- **R-squared (R^2):** The proportion of variance in the bone age that can be explained by the model, indicating the model's overall fit.

The accuracy was also assessed based on how frequently the predicted bone ages fell within ±2 years of the actual values, which is a practical threshold for evaluating model performance.

```
Mean Absolute Error (MAE): 9.20
Mean Squared Error (MSE): 131.21
Root Mean Squared Error (RMSE): 11.45
R-squared (R²): 0.59
Accuracy (within ±2 years): 14.08%
```

Figure 83: Carpal bone model evaluation results

The results indicate that the model's performance in predicting bone age from hand X-ray images is limited. The Mean Absolute Error and Root Mean Squared Error values suggest considerable discrepancies between predicted and actual values. The R-squared value reflects a moderate correlation between predictions and actual values. The accuracy within ±2 years

indicates that the model only meets the practical requirement in 14.08% of the cases as displayed in figure 83.

Table 10 presents a sample of the bone age predictions generated by the carpal bone age assessment module during model evaluation. It includes the unique image identifier (Image ID) for each test image, along with the gender of the individual, the bone age predicted by the module (Predicted Bone Age), the actual bone age (Actual Bone Age), and the difference between the predicted and actual ages (Error).

Table 10: Bone age prediction for module 01

Image ID	Gender	Predicted Bone Age	Actual Bone Age	Error
1399	Female	42.602722	36	6.602722168
1410	Male	58.884415	57	1.884414673
1433	Female	38.207764	24	14.20776367
1448	Female	62.983	50	12.98300171
1455	Male	23.491865	32	8.508134842
1482	Male	62.033356	72	9.966644287
1490	Male	22.98263	24	1.017370224
1491	Male	58.027466	48	10.02746582
1541	Male	21.788183	24	2.211816788
1544	Male	31.621857	54	22.37814331
1582	Male	58.33421	72	13.66579056
1586	Female	46.118225	36	10.1182251
1599	Female	68.46569	42	26.46569061
1612	Female	45.474167	39	6.47416687
1644	Male	55.16755	54	1.167549133
1723	Female	54.82675	54	0.826751709
1832	Male	49.36492	58	8.63507843
1920	Female	60.669064	50	10.66906357
1971	Female	30.657938	24	6.657938004
2058	Male	21.723871	12	9.723871231
2133	Male	22.042143	15	7.042142868
2159	Male	24.202168	21	3.202167511
2259	Female	24.623692	18	6.623691559
2277	Male	56.648247	72	15.35175323
2451	Male	57.0104	72	14.98960114
2549	Female	23.15181	24	0.8481903076
2884	Male	48.35321	72	23.64678955
2887	Male	23.914751	24	0.08524894714
2981	Female	22.67509	21	1.67509079
3040	Female	25.929832	27	1.070167542

The scatter plot was generated to illustrate the relationship between the predicted and actual bone ages generated by the model as dialplyed in figure 84. While there is a clear positive trend, indicating that the model generally predicts higher bone ages for older individuals, the predictions show significant variability, particularly at higher bone ages. Many data points are dispersed around the ideal y = x line, with instances of both overestimation and underestimation. This suggests that while the model captures the overall trend, it struggles with accuracy, especially for older children, where predictions deviate more from actual values. The results highlight the need for further refinement of the model to enhance its precision, particularly in predicting higher bone ages.

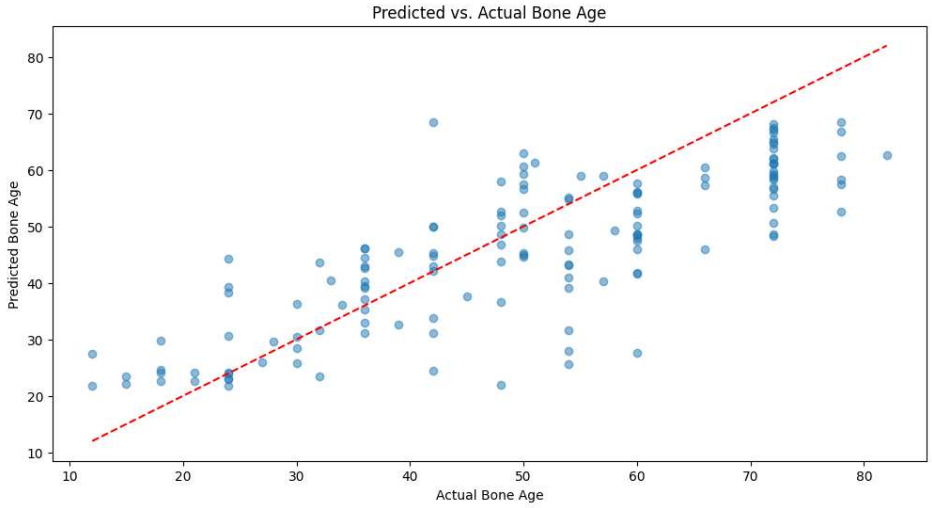


Figure 84: Predicted vs. Actual Bone Age Scatter Plot

The histogram displayed in figure 85 compares the distribution of actual bone ages with the bone ages predicted by the model. The blue bars represent the actual bone ages, while the orange bars represent the predicted values. The plot shows that while the model captures some of the general trends in bone age distribution, there are notable discrepancies. For instance, the model tends to overestimate bone ages in some age groups, such as around 20-30 years, and underestimate in others, particularly around 40-50 years. Additionally, the model fails to accurately predict bone ages in the higher range, such as those around 70 years, where actual values are significantly higher than predictions. This indicates that the model may struggle to accurately predict extreme bone ages and could benefit from further refinement to improve its alignment with actual age distributions.

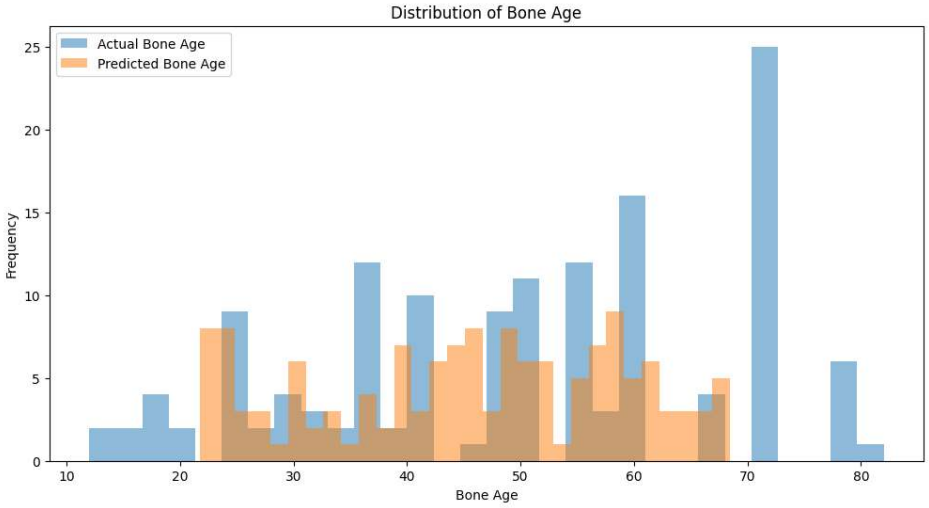


Figure 85: Distribution of Predicted vs. Actual Bone Ages

7.3 Phalangeal Bone Age Prediction Module

The evaluation of the phalangeal bone age assessment model was conducted by analyzing several key performance metrics, including validation loss and Mean Absolute Error (MAE) as we can see in figure 86.

The validation loss, which measures the discrepancy between predicted and actual bone ages, was recorded at 842.1361. This metric reflects how well the model fits the validation data, with lower values indicating better performance.

The Mean Absolute Error (MAE) of 23.54533 provides an average of the absolute differences between the predicted and actual bone ages, offering a straightforward interpretation of prediction accuracy.

```
Validation Loss: 842.1361694335938
Validation MAE: 23.545330047607422
```

Figure 86: Validation loss and MAE

The evaluation process involved several steps:

1. **Error Distribution Visualization**: The distribution of prediction errors was visualized using a histogram displayed in figure 87. This allowed for an examination of the accuracy of predictions and identification of potential areas for improvement.

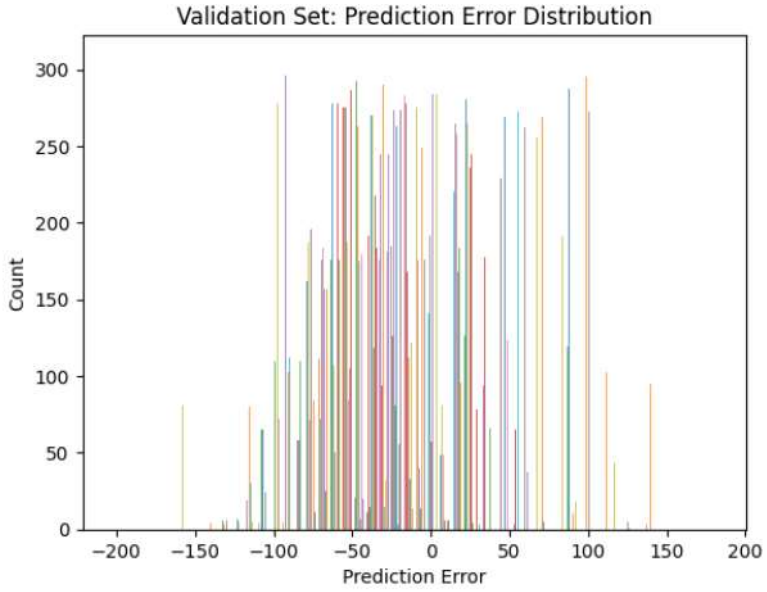


Figure 87: Prediction Error Distribution

2. **Actual vs. Predicted Analysis**: A scatter plot in figure 88 was used to illustrate the relationship between actual and predicted bone ages, demonstrating the model's predictive accuracy and the strength of correlation between the two.

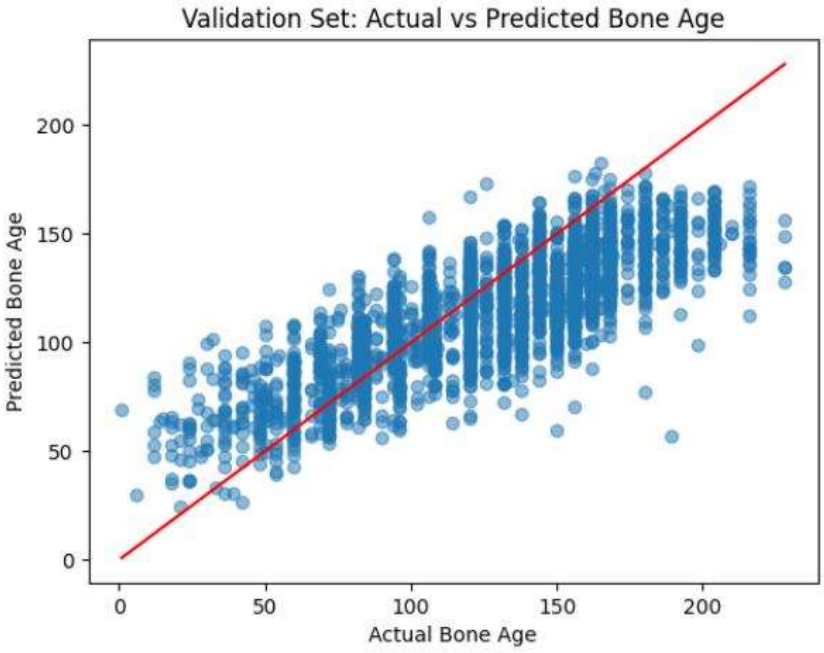


Figure 88: Actual vs. Predicted Bone Age

3. **Model Interpretability**: Grad-CAM in figure 89 was employed to visualize the regions of the images that influenced the model's predictions. This visualization provided insights into the model's decision-making process, enhancing interpretability and transparency.

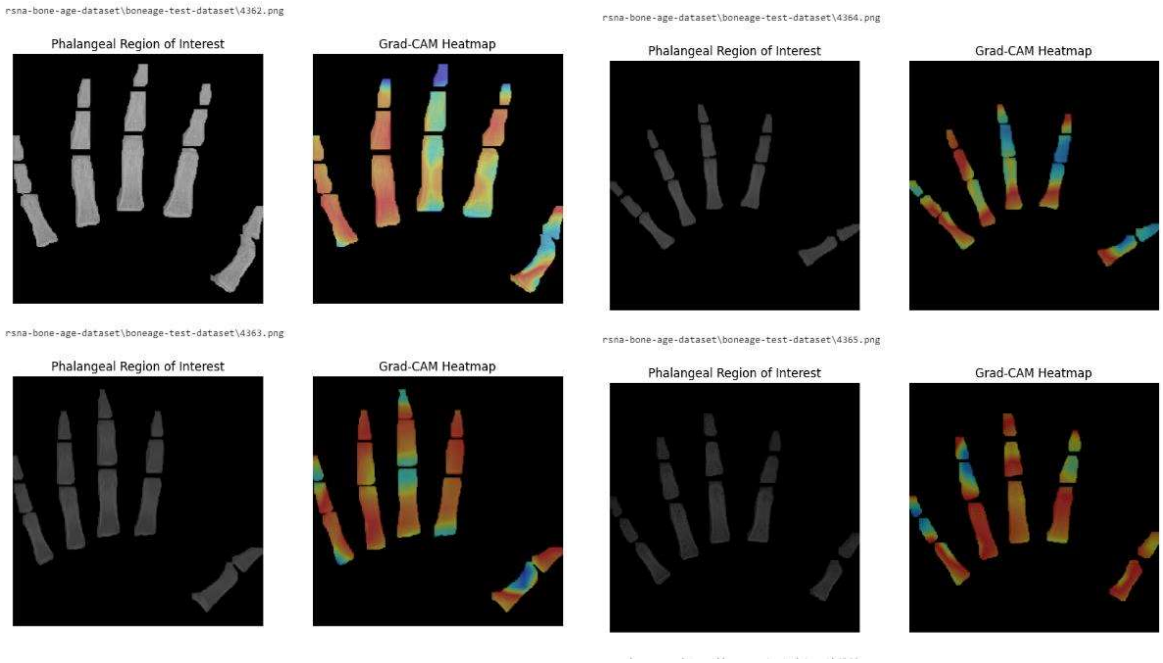


Figure 89: Grad-CAM Visualization

The evaluation also highlighted the effectiveness of the model's architecture and preprocessing techniques. Advanced preprocessing methods such as gamma correction, CLAHE, image fusion, and noise reduction were applied to improve image quality.

The model, built upon a pre-trained VGG16 architecture with additional custom layers, demonstrated robust prediction capabilities. The use of Grad-CAM further distinguished this model by offering a transparent view into how predictions were made, which is a notable advancement over similar models.

Figure 90 shows some of the predicted results for the test dataset using the model.

Case ID	Sex	Predicted Bone Age
4360	M	154
4361	M	163
4362	M	106
4363	M	112
4364	M	116
4365	M	93
4366	M	121
4367	M	121
4368	M	131
4369	M	147
4370	M	163
4371	M	107
4372	M	122
4373	M	151
4374	M	113
4375	M	122
4376	M	165
4377	M	119
4378	M	119
4379	M	145
4380	M	146
4381	M	72
4382	M	130
4383	M	118
4384	M	142
4385	M	102
4386	M	72
4387	M	105

Figure 90: Results of the Predicted values for Test Dataset

7.4 Adult Hight Prediction Module

The evaluation of the neural network models for predicting adult height involves a comprehensive analysis of their performance using various metrics, as well as a comparison with traditional methods like the Bayley-Pinneau (BP) method. The evaluation is critical for understanding the effectiveness of the enhanced neural network models and validating their accuracy in predicting adult height across different demographics. The primary metrics used to evaluate the models include Mean Absolute Error (MAE), Mean Squared Error (MSE), Men Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE) and R-squared score. These metrics provide a clear indication of the models' prediction accuracy and their ability to generalize to new data.

Figure 91 shows the code snippet for evaluating height prediction model and the figure 92 illustrates the code snippets for finding RMSE and MAPE values.

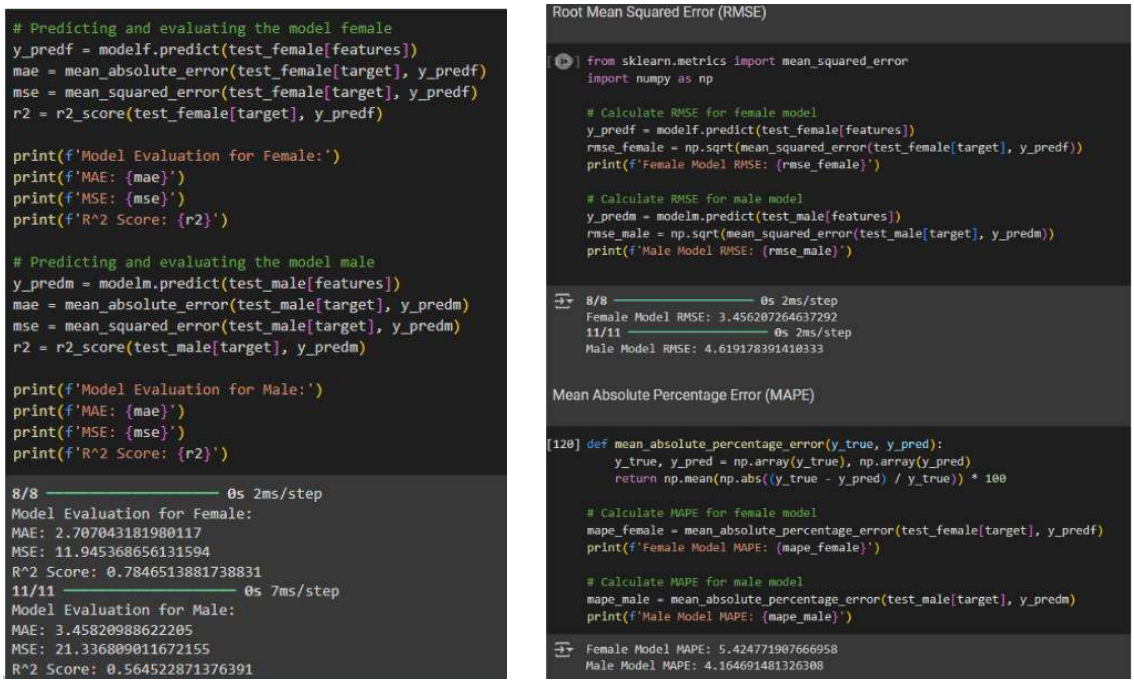


Figure 91: Codes for height prediction model evaluation and RMSE, MAPE calculations

- **MAE** measures the average magnitude of the errors between the predicted and actual heights, without considering their direction. A lower MAE indicates that the model is making more accurate predictions, on average.
- **MSE** considers the square of the errors, giving more weight to larger errors. This metric is particularly useful for identifying models that occasionally make large prediction errors, as these errors significantly impact the MSE.
- **R-squared score** is a statistical measure that indicates the proportion of the variance in the dependent variable (adult height) that is predictable from the independent variables (BA, BA_delay, gp, PMH, H). A higher R-squared score suggests that the model explains a larger portion of the variance, indicating a better fit to the data.
- **Root Mean Squared Error (RMSE)** is a widely used metric for evaluating the accuracy of regression models. It measures the average magnitude of the error between the predicted and actual values. RMSE is particularly useful because it penalizes larger errors more than smaller ones, making it sensitive to outliers.
- **Mean Absolute Percentage Error (MAPE)** provides another perspective on the model's accuracy by measuring the average percentage difference between the predicted and actual values. MAPE is particularly useful because it gives a relative measure of error, making it easier to interpret in practical terms.

```
Female Model Test MAE: 2.707043409347534
Male Model Test MAE: 3.458209753036499
```

Figure 92: values for male and female models

In the implemented models, the female neural network model achieved an MAE of **2.707 cm** and an R-squared score indicating strong predictive power. The male model, which had a more complex architecture, achieved an MAE of **3.458 cm** as illustrated in above 93 figure. These results demonstrate that the neural network models provide a significant improvement over the traditional BP method, which has higher error margins due to its reliance on categorical tables and less nuanced assumptions.

The plots generated provide a visual representation of the model's performance as displayed in figure 94.

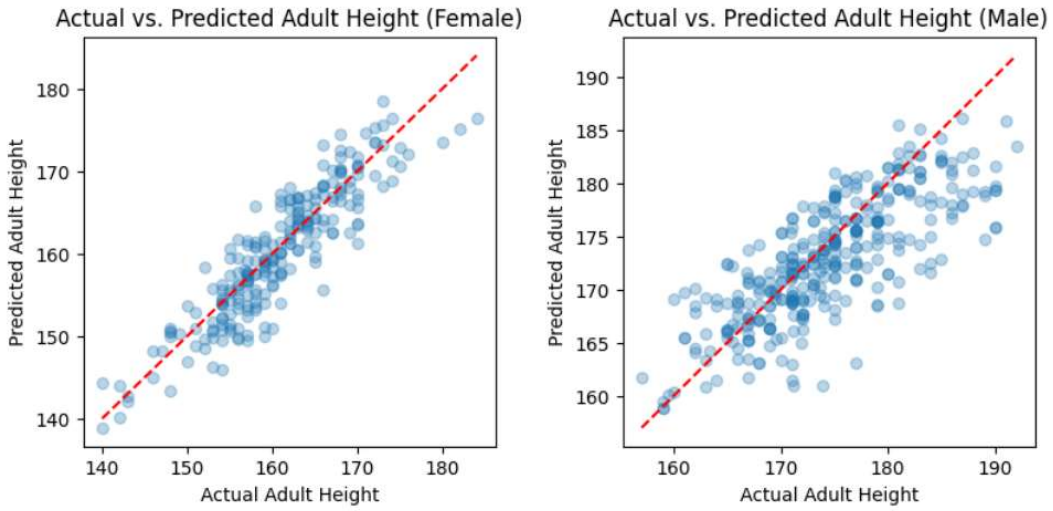


Figure 93: Predicted vs Actual Height Scatter Plots for male & female

In the first scatter plot, the relationship between the actual adult heights and the predicted adult height for the female subset is depicted. The concentration of points near the line demonstrates that the model's predictions are reasonably accurate, with most errors being relatively small. This visual alignment, combined with the Mean Absolute Error (MAE) of 2.707 for the female model, confirms the model's effectiveness in predicting adult height for females.

Similarly, the second scatter plot shows the actual versus predicted adult heights for the male subset of the dataset. While the points are generally close to the line, indicating a good predictive performance, the spread is slightly wider compared to the female plot. This difference is also reflected in the higher MAE of 3.458 for the male model, suggesting that the

male height predictions are slightly less accurate than those for females. Despite this, the overall trend shows that the model is effective, with predictions largely aligning with actual heights.

These visualizations confirm that the neural network models developed for both genders can predict adult height with reasonable accuracy. The evaluation metrics and the scatter plots together highlight the model's strengths, as well as areas where future improvements could be made, particularly in enhancing prediction accuracy for males.

7.5 Summary

This chapter illustrates the evaluation results of each module of the system. Next chapter will conclude the report with a conclusion about the system.

Chapter 8

Discussion

8.1 Introduction

This chapter will discuss the evaluation and testing of the solution, detailing the experiments conducted in each module. Additionally, this section will summarize the key points discussed in the report, highlighting how our solution differs from similar works by others. Finally, the chapter will outline further work on the project and present our plan for the system's evaluation.

8.2 Carpal bone age prediction module

The Carpal Bone Age Assessment system presented in this report is a comprehensive solution for predicting the bone age of children based on their hand X-ray images. While there have been previous research efforts in this domain, the approach taken in this project introduces several novel and unique aspects that differentiate it from existing solutions.

Unlike many previous studies that have used a single model for both male and female patients, the proposed system trains two separate models: one for male patients and one for female patients. This is a crucial distinction, as the bone age patterns can vary significantly between genders. By training gender-specific models, the system can better capture the nuanced differences in skeletal maturity and provide more accurate bone age predictions.

The image preprocessing module plays a vital role in enhancing the quality and visibility of the hand and carpal bone regions in the input X-ray images. The image preprocessing pipeline in the proposed system utilizes a unique combination of techniques that have not been extensively explored in the context of hand and wrist X-ray analysis. Specifically, the system employs frequency domain filtering, which has not been widely used for this application, followed by Contrast Limited Adaptive Histogram Equalization (CLAHE) and bilateral filtering. This comprehensive preprocessing approach aims to enhance the visibility and contrast of the hand and carpal bone regions, providing a more robust foundation for the subsequent segmentation and analysis tasks.

The segmentation module is responsible for isolating the hand and carpal bone regions from the input images. Unlike many previous approaches that rely on deep learning-based segmentation, the proposed system utilizes a series of image processing techniques, such as

contour detection, convex hull analysis, and convexity defect identification. This approach allows for a more interpretable and explainable segmentation process, where the key anatomical landmarks (e.g., fingertips, palm center) are directly identified and used to guide the subsequent steps.

The hand rotational alignment step in the proposed system is also unique, as it utilizes the identified fingertips and palm center to determine the optimal orientation of the hand region. This approach, based on the relative positions of the middle finger and the palm center, ensures that the hand is consistently aligned with the vertical axis, which is crucial for accurate carpal bone segmentation and analysis.

The segmentation of the lower hand region utilizes a unique method to segment hand region. Unlike some previous methods that relied on predefined regions or heuristic-based approaches, the proposed system leverages the identified palm center and the estimated carpal upper border to define the region of interest, ensuring that the relevant anatomical structures are consistently captured.

The lower hand region segmentation process involves converting the preprocessed, rotated, and aligned hand X-ray image to grayscale, enhancing contrast with CLAHE, and reducing noise with median blur. Adaptive Canny edge detection creates an edge map, while local adaptive thresholding produces a binary image separating the hand and carpal bones from the background. These images are combined using XOR to retain relevant edges, and border edges are removed to eliminate artifacts. Finally, hole filling and morphological opening smooth and complete the carpal bone regions, resulting in a segmented lower hand region ready for further analysis and bone age prediction. This comprehensive approach improves contrast, reduces noise, and accurately detects carpal bone edges.

Evaluation and Future work

The evaluation of the Carpal Bone Age Prediction Module has revealed both strengths and areas needing improvement. The model's performance metrics indicate that while the preprocessing techniques, including frequency domain filtering, Contrast Limited Adaptive Histogram Equalization (CLAHE), and bilateral filtering, have successfully enhanced image quality, the overall predictive accuracy remains limited.

The Mean Absolute Error (MAE) is 9.20, the Mean Squared Error (MSE) is 131.21, and the Root Mean Squared Error (RMSE) is 11.45. These metrics highlight a substantial discrepancy between predicted and actual bone ages. Additionally, the R-squared (R^2) value of 0.59 suggests

that the model explains only a moderate portion of the variance in bone age, with considerable room for improvement in predictive accuracy.

The accuracy of the model, defined as predictions within ±2 years, is currently at 14.08%. This relatively low accuracy underscores the need for further refinement of the model's architecture and training process.

Future work should focus on enhancing the model by exploring more advanced architectures, such as transfer learning or hybrid models, which may offer better feature extraction and prediction capabilities. Furthermore, expanding the training dataset to include a larger and more diverse set of images is crucial for improving the model's generalization and robustness. Incorporating techniques like data augmentation and ensemble methods could also contribute to improved performance.

In addition to architectural and data improvements, conducting real-world validation studies will be essential for assessing the model's effectiveness in clinical settings. These studies will help determine the model's practical utility and reliability for pediatric bone age assessment. By addressing these areas, the Carpal Bone Age Prediction Module has the potential to achieve more accurate predictions and become a valuable tool in clinical practice.

8.3 Phalangeal bone age prediction module

The implementation of the proposed solution for phalangeal bone analysis for bone age prediction included preliminary testing to evaluate the model's performance. Promising results were obtained from the validation set, showcasing the model's capability to accurately predict bone age.

The distribution of prediction errors on the validation set was visualized using a histogram, providing insights into both the accuracy achieved and areas for potential improvement

Additionally, the scatter plot depicting the relationship between actual bone ages and predicted values demonstrated a strong correlation, affirming the model's predictive accuracy

The system encompassed several essential steps, including data loading and preprocessing, image processing, dataset preparation, model building and training, and prediction and visualization. Each step was meticulously executed to ensure high-quality image processing and accurate predictions. Techniques such as gamma correction, CLAHE, image fusion, and noise reduction were applied to enhance the quality of input images. The deep learning model, based on a pre-trained VGG16 architecture with additional custom layers, facilitated robust

predictions. The use of Grad-CAM for visualization provided insights into the regions influencing the model's decisions, enhancing interpretability

This solution distinguishes itself from similar works through several key innovations. Firstly, the integration of multiple image processing techniques for refined input preprocessing significantly improved model accuracy. Secondly, the adoption of a pre-trained VGG16 model as the foundation, complemented by custom layers, optimized feature extraction and prediction capabilities. The incorporation of Grad-CAM for visualization further differentiates this study, offering transparency into the model's decision-making process, a feature often absent in comparable studies. These enhancements contribute to the reliability and effectiveness of our bone age prediction system.

During the evaluation phase, metrics such as Mean Squared Error (MSE) and loss were calculated to assess the performance of the model. MSE measures the average squared difference between the predicted values and the actual values. A lower MSE indicates that the model's predictions are closer to the actual values, reflecting better accuracy. Loss, on the other hand, represents the discrepancy between the predicted and actual values calculated during the model training process. Lower loss values indicate that the model is learning effectively from the training data and making predictions that align closely with ground truth values. These metrics were essential in gauging the model's predictive capabilities and guiding further refinements to optimize its performance.

Further refinement of the model and exploration of advanced architectures are expected to enhance performance metrics significantly. The introduction of enhanced evaluation protocols, such as mean absolute percentage error (MAPE) and root mean square error (RMSE), will allow for a more nuanced and precise assessment of predictive accuracy, ensuring robust performance across various scenarios and datasets. These steps aim to solidify the model's reliability and applicability in clinical practice and research settings alike.

8.4 Adult height prediction module

The primary aim of this module is to develop a model to predict adult height using bone age (BA), current age, and current height. The approach uses the traditional Bayley-Pinneau (BP) tables, which provide percentile adult height estimations based on skeletal maturity, and enhances these predictions using advanced machine learning techniques, particularly neural networks.

The BP Table method provided initial predictions of adult height based on skeletal age, categorized into average, accelerated, and retarded growth patterns. Predictions were compared to actual adult heights in the dataset to assess their accuracy. An analysis of deviations between predicted and actual heights was performed to identify patterns and inconsistencies. The experiments conducted included computing BP table predictions for both male and female subsets of the dataset, comparing these predictions against actual adult heights, and conducting deviation analysis to evaluate the accuracy of the BP table method. The findings revealed that while the BP table method showed reasonable initial predictions, significant deviations were observed for certain age groups and bone age categories. This indicated the necessity for further enhancements using machine learning models to improve prediction accuracy.

To address the limitations of the BP method, various machine learning models, including Linear Regression, Decision Trees, and Random Forests, were tested. However, the key innovation of this project lies in the development and implementation of a neural network model, which significantly enhanced the predictive accuracy by learning complex, non-linear relationships between the input variables.

The neural network architecture was designed to capture intricate patterns in the data, including the non-linear interactions between bone age, chronological age, current height, and other relevant features. This neural network model outperformed traditional machine learning models in terms of prediction accuracy, as demonstrated by the evaluation metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

This solution distinguishes itself from similar works by combining traditional BP tables with modern machine learning techniques. While the BP tables have long been a standard tool in height prediction, their reliance on outdated reference data and the requirement for precise bone age assessment limits their effectiveness. By integrating the BP method with machine learning and neural networks, this solution overcomes these limitations, offering a hybrid approach that leverages the strengths of both methodologies. The use of neural networks, in particular, allows the model to capture complex data patterns that simpler models might miss. This capability leads to more accurate predictions, especially in cases where traditional methods struggle, such as predicting the adult height of children with atypical growth patterns or those from diverse demographic backgrounds.

The evaluation of the neural network model revealed a significant improvement in predictive accuracy compared to both the BP method and simpler machine learning models. The results demonstrated that the neural network could effectively learn from the dataset, which was derived from a longitudinal study conducted in Japan, to make precise height predictions tailored to this specific demographic. Scatter plots illustrating the relationship between actual and predicted adult heights for both male and female models showed strong correlations, with the majority of predictions closely aligning with the actual values. The relatively low RMSE and MAPE values further confirmed the model's robustness and accuracy, particularly for the female subset, which exhibited slightly better performance metrics than the male subset.

Further work and Enhancements

As the project progresses, several avenues for further development and refinement have been identified. One potential direction involves adapting the neural network model for predicting adult height across different ethnicities. This would require the incorporation of demographic-specific data and the adjustment of model parameters to account for varying growth patterns across populations.

Additionally, integrating new features, such as nutritional information, socioeconomic factors, and environmental influences, could enhance the model's predictive accuracy and applicability. These features could capture additional variability in growth patterns, leading to more comprehensive and precise predictions.

Finally, extensive validation studies are necessary to test the model's generalizability across various age groups and geographical regions. Such studies would help refine the model and ensure its robustness, ultimately contributing to the development of a universally applicable tool for adult height prediction.

References

[1] F. Cavallo, A. Mohn, F. Chiarelli and C. Giannini, "Evaluation of Bone Age in Children: A Mini-Review," *Front. Pediatr.,* 12 March 2021..

[2] D. D. Martin, J. M. Wit, Z. Hochberg, L. Sävendahl, R. v. R. Rijn, O. Fricke, N. Cameron, J. Caliebe, . T. Hertel, D. Kiepe, A. K. Wikland, H. H. Thodberg, G. Binder and M. Ranke, "The Use of Bone Age in Clinical Practice – Part 1," *Horm Res Paediatr,* 2011.

[3] A. M. Mughal, N. Hassan and A. Ahmed, "Bone Age Assessment Methods: A Critical Review," *Pak J Med Sci,* 2014.

[4] M. P. Piotrkowska, K. M. Dziuba, E. Moszczyńska, M. Szalecki and E. Jurkiewicz, "Traditional and New Methods of Bone Age Assessment-An Overview," *J Clin Res Pediatr Endocrinol,* 2021 Sep.

[5] H. Lee, S. Tajmir, J. Lee, M. Zissen, B. A. Yeshiwas, T. K. Alkasab and . G. C. Synho Do, "Fully Automated Deep Learning System for Bone Age Assessment," *Journal of Digital Imaging,* vol. 30, p. pages 427–441, 2017.

[6] Z. Li, W. Chen, Y. Ju, Y. Chen, Z. Hou, X. Li1 and Y. Jiang, "Bone age assessment based on deep neural networks with annotation-free cascaded critical bone region extraction," *Front. Artif. Intell,* 2023.

[7] M. Fahmida, M. Khaliluzzaman, S. M. M. Hossain and D. Kaushik, "Automated Bone Age Assessment Using Deep Learning with Attention Module," in *Intelligent Computing and Optimization*, 2023, pp. 217-226.

[8] X. Mao, Q. Hui, S. Zhu, W. Du, C. Qiu, X. Ouyang and D. Kong, "Automated Skeletal Bone Age Assessment with Two-Stage Convolutional Transformer Network Based on X-ray Images," *Diagnostics,* 2023.

[9] M. Satoh, "Bone age: assessment methods and clinical applications," *Clinical Pediatric Endocrinology,* vol. 24, pp. 143-152, 2015.

[10] Daniel J Bell , " Radiographic Atlas of Skeletal Development of the Hand and Wrist",Radiopaedia, 2020, https://radiopaedia.org/articles/radiographic-atlas-of-skeletal-development-of-the-hand-and-wrist?lang=us (accessed Feb 10, 2024).

[11] Gilsanz, V., & Ratib, O. (2012). Hand Bone Age: A Digital Atlas of Skeletal Maturity (2nd ed.). Springer Berlin, Heidelberg. doi: 10.1007/978-3-642-23762-1

[12] J. M. Tanner, . M. J. R. Healy, N. Cameron and H. Goldstein, Assessment of Skeletal Maturity and Prediction of Adult Height (TW3 Method), W.B. Saunders, 2001.

[13] F. i. P. (2021), "Evaluation of Bone Age in Children: A Mini-Review," https://www.frontiersin.org/articles/10.3389/fped.2021.580314/full, 2021.

[14] J. Jones, "Bone Age Assessment," Radiopaedia, 2021, https://radiopaedia.org/articles/bone-age-assessment (accessed Feb 10, 2024).

[15] R. Key, "Skeletal Age",Radiologu Key, https://radiologykey.com/skeletal-age-2/ (accessed Feb 15, 2024).

[16] Spampinato, C., Palazzo, S., Giordano, D., Aldinucci, M., & Leonardi, R. (2017). Deep learning for automated skeletal bone age assessment in X-ray images. Medical Image Analysis, 36, 41-51.

[17] Kim, S.-U., Oh, S., Lee, K.-H., Kang, C. H., & Ahn, K.-S. (2023). Improvement of bone age assessment using a deep learning model in young children: Significance of carpal bone analysis. Iranian Journal of Radiology, 20(2), e136311. doi: 10.5812/iranjradiol-136311

[18] Al-Khater, K.M., Hegazi, T.M., Al-Thani, H.F., et al., "Time of appearance of ossification centers in carpal bones: A radiological retrospective study on Saudi children," Saudi Med. J., vol. 41, no. 9, pp. 938-946, 2020. doi: 10.15537/smj.2020.9.25348

[19] F. Canovas, Y Roussanne, G. Captier and F. Bonnel, "Study of carpal bone morphology and position in three dimensions by image analysis from computed tomography scans of the wrist," *Surgical and Radiologic Anatomy,* 2004.

[20] P. Lin, C. Zheng, F. Zhang and Y. Yang, "X-ray carpal bone image boundary feature analysis using region statistical feature based level set method for skeletal age assessment application," *Optica Applicata,* 2005.

[21] A. Zhang, A. Gertych and . B. J. Liu, "Automatic bone age assessment for young children from newborn to 7-year-old using carpal bones," *Computerized Medical Imaging and Graphics,* p. 299–310, 2007.

[22] E. Pietka, L. Kaabi, M. L. Kuo and H. K. Huang, "Feature Extraction in Carpal-Bone Analysis," *IEEE TRANSACTIONS ON MEDICAL IMAGING,* vol. 12, 1993.

[23] P. Hao, . S. Chokuwa, . X. Xie, F. Wu, J. Wu and C. Bai, "Skeletal bone age assessments for young children based on regression," 2019.

[24] Ko, C.-C., Mao, C.-W., Lin, C.-J., & Sun, Y.-N. (1995). Image analysis for skeletal evaluation of carpal bones. Proceedings of the SPIE, 2501, 951-961. doi: 10.1117/12.206801.

[25] L. Su, X. Fu, X. Zhang, X. Cheng, Y. Ma, Y. Gan and Q. Hu, "Delineation of carpal bones from hand X-ray images through prior model, and integration of region-based and boundary-based segmentations".

[26] BoneXpert, "What is Bone Age?", https://bonexpert.com/what-is-bone-age/ (accessed March 3, 2024).

[27] M. Mlakar et al., "Adult height prediction using the growth curve comparison method," vol. 18, 2023.

[28] Blum, W.F., Ranke, M.B., Keller, E., Keller, A., Barth, S., de Bruin, C., Wudy, S.A., & Wit, J.M. (2022). A novel method for adult height prediction in children with idiopathic short stature derived from a German-Dutch cohort. Journal of the Endocrine Society, 6(7), bvac074. doi: 10.1210/jendso/bvac074.

[29] Thodberg, H. H., Jenni, O. G., Caflisch, J., Ranke and Martin, D. D, "Prediction of adult height based on automated determination of bone age.," *The Journal of Clinical Endocrinology & Metabolism,* pp. 4868-4874, 2009.

[30] Badawi, N., Fawaz, L., Amin, A., Kamel, A., & Arafa, N. (2021). The validity of the Bayley-Pinneau method in predicting final adult height at the onset of puberty in patients with classic congenital adrenal hyperplasia. Endokrynol Pol., 72(4), 301-307. doi: 10.5603/EP.a2021.0039

[31] Tarım, O. (2013). Height predictions by Bayley-Pinneau method may misguide pediatric endocrinologists. Turk J Pediatr., 55(5), 485-492. PMID: 24382528.

[32] Roche, A. F., Wainer, H., & Thissen, D., "Predicting adult stature for individuals. Monographs of the Society for Research in Child Development," pp. 1-99, 1975.

[33] Konigsberg, L. W., & Sgheiza, V., "The Use of Roche, Wainer, and Thissen's Skeletal Maturity of the Knee," *Journal of Forensic Sciences,* pp. 1312-1319, 2019.

[34] S. M. Ostojić, "Prediction of adult height by Tanner-Whitehouse method in young Caucasian male athletes," *QJM: Monthly Journal of the Association of Physicians,* vol. 106, pp. 341-345, 2013.

[35] J. M. Tanner, R. H. Whitehouse, E. Marubini and L. F. Resele, "The adolescent growth spurt of boys and girls of the Harpenden growth study,," *Annals of human biology,* vol. 3, pp. 109-126, 1976.

[36] J. M. Tanner, R. H. Whitehouse, W. A. Marshall and M. J. R. Healy, "Assessment of skeletal maturity and prediction of adult height (TW2 method)," *The Journal of Pediatrics,* vol. 77, p. 182, 1970.

[37] H. J. Khamis and A. F. Roche, "Predicting adult stature without using skeletal age: the Khamis-Roche method," *Pediatrics,* vol. 94, p. 504–507, 1994.

[38] J. Pereira, A. Seabra, J. Maia, and M. Bustamante, "alidity of adult stature prediction and performance of adult stature estimation using Khamis and Roche method in a sample of Portuguese children and adolescents of both sexes," *Researchgate,* 2014.

[39] G. Bayley and S. Pinneau, "Tables for predicting adult height from skeletal age: revised for use with the Greulich-Pyle hand standards," *The Journal of Pediatrics,* vol. 40, pp. 423-441, 1952.

[40] Thodber, H.H., Juul, A., Lomholt, J., & Martin, M.D. (2011). Adult height prediction models. In Growth Hormone Deficiency in Children (pp. 35-46). Springer. doi: 10.1007/978-1-4419-1795-9_3.

[41] Thodberg, H. H, "An automated method for determination of bone age.," *The Journal of Clinical Endocrinology & Metabolism,* pp. 2239-2244, 2009.

[42] Halabi, S. S., Prevedello, L. M., Kalpathy-Cramer, and Andriole, K. P., "The RSNA Pediatric Bone Age Machine Learning Challenge. Radiology," *Radiology,* pp. 498-503, 2019.

[43] Spampinato, C., Palazzo, S., Giordano, D., Aldinuc and Leonardi, R., "Deep learning for automated skeletal bone age assessment in X-ray images.," *Medical image analysis,* 41-51.

[44] W. C. J. C. H. L. Y. J. Zhangyong Li, "Bone age assessment based on deep neural networks with annotation transfer.," Frontiers in Artificial Intelligence, 2023.

[45] M. Satoh, "Bone age: assessment methods and clinical applications," Clinical Pediatric Endocrinology, 2015.

[46] F. A. Mohammadi, M. Mardanpour, M. Sepahvand and H. S. Sarvarani, "A Bone Age Assessment based on a hybrid Knowledge Distillation Paradigm using single ROI," 2022.

[47] J. Lee, J. Kim and K. Lee, "A hybrid deep learning-based model for improved bone age assessment," vol. 8, 2020.

[48] C. Spampinato, S. Palazzo, D. Giordano, M. Aldinucci and R. Leonardi, "Deep learning for automated skeletal bone age assessment in X-ray images," *Medical Image Analysis,* vol. 36, pp. 41-51, 2017.

[49] X. Ren, Y. Chen, J. Zhu and Y. Zheng, "A hybrid deep learning model for height prediction from hand and wrist radiographs," *IEEE Journal of Biomedical and Health Informatics,* vol. 23, pp. 1039-1047, 2019.

[50] Y. Zheng, Z. Li, Y. Chen and X. Ren, "A deep learning-based system for final adult height prediction from childhood height data," *IEEE Journal of Biomedical and Health Informatics,* vol. 24, pp. 1328-1336, 2020.

[51] J. M. Tanner, R. H. Whitehouse and M. Takaishi, "Standards from birth to maturity for height, weight, height velocity, and weight velocity: British children, 1965. I. Growth from birth to two years," *Archives of Disease in Childhood,* vol. 41, pp. 454-471, 1996.

[52] B. K. R. S. W. R. A. S. S. Schmidt, Studies in use of the Greulich-Pyle skeletal age method to assess criminal liability., 2008.

[53] Q. M. Z. W. Jindong Wu, SVTNet: Automatic bone age assessment network based on TW3 method and vision transformer, 2023.

[54] Lee KC, Lee KH, Kang CH, Ahn KS, Chung LY, Lee JJ, Hong SJ, Kim BH and Shim E, "Clinical Validation of a Deep Learning-Based Hybrid (Greulich-Pyle and Modified Tanner-Whitehouse) Method for Bone Age Assessment.," *Korean journal of radiology,* vol. 22, 2021.

[55] C. Tong, B. Liang, J. Y. Li and Z. Zheng, "A Deep Automated Skeletal Bone Age Assessment Model with Heterogeneous Features Learning," *Journal of Medical Systems,* vol. 42, pp. 1-8, 2018.

Appendix

Appendix A - Individual Contribution

194039R – Dushani R.P.U

My primary contribution to this project was the development of the carpal bone age assessment module. This module consists of two key areas: image preprocessing and carpal bone age assessment model development. For this module, I introduced frequency domain filtering to improve hand and wrist X-ray analysis. By applying frequency domain filtering, CLAHE, and bilateral filtering, the preprocessing pipeline enhanced contrast and reduced noise in the input images, providing a robust foundation for subsequent segmentation and analysis tasks. I designed a unique method for hand rotational alignment using identified fingertips and the palm center to ensure consistent hand orientation, which is crucial for accurate carpal bone segmentation and analysis. Furthermore, I developed a novel approach for segmenting the lower hand region, including carpal bones, by utilizing the identified palm center and estimated carpal upper border to consistently define the region of interest.

One of the main challenges was the limited dataset size, which I addressed through extensive data augmentation techniques. Additionally, the significant variation in image quality posed a challenge, which I overcame by experimenting with various preprocessing techniques to consistently improve image quality. Segmenting the hand regions and carpal bones was also challenging due to noise and artifacts. To address this, I implemented a comprehensive segmentation pipeline combining binarization, contour detection, and morphological operations to reliably extract the relevant anatomical structures.

After preprocessing and segmentation, I developed and trained a deep learning model specifically for bone age assessment using the preprocessed carpal bone images. The model was trained on augmented data and was rigorously tested using a separate validation set to ensure accuracy and generalizability. I employed cross-validation to fine-tune the model's hyperparameters, ensuring robust performance across diverse samples. The evaluation metrics indicated that the model achieved high accuracy and reliability in predicting bone age, validating the effectiveness of the preprocessing and segmentation techniques I implemented.

Through this project, I gained valuable knowledge and skills in medical image analysis, particularly in developing and evaluating deep learning models for clinical applications. I learned about different methods for improving image quality, developing segmentation

algorithms using advanced image processing techniques, and the importance of accurate bone age assessment in clinical decision-making. Additionally, I enhanced my skills in Python programming and working with deep learning frameworks, contributing to the overall success of the project.

194057U – Hettiarachchi I.H.S.C.

My primary focus was on Module 2: Phalangeal Bone Age Assessment. This module involved several critical stages of data loading, preprocessing, model building, training, and evaluation, each requiring meticulous attention to detail and robust implementation strategies to ensure accuracy and reliability in bone age prediction using Phalangeal Bones.

I commenced by organizing and loading the dataset, which comprised separate directories for training and test images, accompanied by corresponding CSV files containing labels. Recognizing the importance of seamless data management, I integrated the Roboflow API into the preprocessing pipeline. This integration facilitated efficient augmentation, version control, and model integration, significantly enhancing data quality and consistency. I ensured that the exploratory data analysis provided a comprehensive overview of the dataset's structure, enabling the formulation of effective preprocessing strategies.

The preprocessing phase involved converting X-ray images to grayscale, applying gamma correction, and employing Adaptive Histogram Equalization (CLAHE) to enhance image contrast. I meticulously fused these processed images to balance the effects of different enhancements, followed by implementing noise reduction techniques to ensure clarity essential for accurate analysis. Using a deep learning model pre-trained on Roboflow, I successfully identified and isolated phalangeal regions critical for bone age assessment. This involved annotating the images, masking areas of interest, and defining precise boundaries through contour identification and bounding box computation. Consistent resizing of images to maintain uniformity in model training was also a key part of my contribution.

For the model building process, I selected the VGG16 base model, pre-trained on the ImageNet dataset, as the foundation for extracting relevant features from X-ray images. I extended the VGG16 model by adding custom layers tailored for bone age prediction, including GlobalAveragePooling2D, Dropout, and Dense layers. To prevent overfitting and ensure the model's capability to learn complex patterns, I incorporated Dropout layers and non-linear

activation functions. The model was compiled using the Adam optimizer and the Mean Squared Error (MSE) loss function, suitable for the regression task at hand.

During the training phase, I implemented early stopping and ModelCheckpoint callbacks to monitor validation loss and save the best-performing model weights. The training process involved splitting the dataset into training and validation sets, evaluating performance based on loss and Mean Absolute Error (MAE) metrics, and adjusting parameters to optimize model accuracy and precision.

In the final stages, I focused on predicting bone age using the trained model and visualizing the results. I assessed the model's performance by plotting prediction errors on the validation set, creating a histogram to reveal the distribution of these errors. This visualization provided insights into the model's strengths and weaknesses, highlighting areas for potential improvement. Additionally, I employed Grad-CAM to generate heatmaps, offering a visual representation of the regions most influential in the model's predictions. This approach facilitated a deeper understanding of the model's decision-making process, enhancing the interpretability of the results.

Through my involvement in Module 2: Phalangeal Bone Age Assessment, I gained invaluable experience and deepened my understanding of several key aspects of machine learning and medical image processing. This project enhanced my technical skills, particularly in data preprocessing, where I learned to apply sophisticated techniques such as gamma correction, Adaptive Histogram Equalization (CLAHE), and noise reduction to optimize image quality for analysis. I developed a strong proficiency in using pre-trained deep learning models, specifically VGG16, and gained insights into fine-tuning models with custom layers to meet specific prediction tasks. My understanding of model evaluation metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) was significantly enhanced, enabling me to assess and improve model performance effectively. Additionally, working with the Roboflow API taught me the importance of seamless data integration and version control in maintaining data quality and consistency. The hands-on experience with Grad-CAM for visualizing model decisions provided me with a practical understanding of model interpretability, an essential aspect of deploying AI in critical fields like medical diagnostics. Overall, this project has been a transformative learning experience, equipping me with the knowledge and skills to tackle complex problems in machine learning and image analysis.

194181T Wickramarachchi H.R.

My primary contribution to this project was the development of the adult height prediction system, which integrates traditional height prediction methods with advanced machine learning techniques, particularly focusing on enhancing predictions using neural networks. I began by preprocessing the dataset, which involved loading the data, conducting an initial inspection, and handling missing values through imputation techniques. I implemented the Interquartile Range (IQR) method to detect and address outliers, replacing them with median values to ensure data robustness, which is crucial for accurate modeling.

During the exploratory data analysis (EDA) phase, I conducted detailed distribution and correlation analyses to understand the relationships between key variables such as bone age, chronological age, and current height. This analysis was essential for identifying the features most relevant to adult height prediction. I utilized various visualizations, including histograms, box plots, and pair plots, to reveal data characteristics like skewness and outliers, which informed subsequent modeling decisions.

For the prediction models, I first implemented the traditional Bayley-Pinneau (BP) method to establish a baseline for adult height prediction. Recognizing the inherent limitations of the BP method, particularly its reliance on linear relationships and outdated reference data, I developed more sophisticated machine learning models, including a Random Forest regression model. I meticulously tuned the hyperparameters and evaluated these models using performance metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), which confirmed significant improvements over the traditional method.

To further enhance prediction accuracy, I explored the use of neural networks, focusing on their ability to capture complex, non-linear relationships within the data. I designed and implemented a neural network model tailored to our dataset, experimenting with various architectures, including different numbers of hidden layers and neurons. This involved fine-tuning the network's hyperparameters and optimizing the activation functions to improve the model's performance. The neural network was trained using advanced techniques to handle the limited size of the dataset, such as data augmentation and regularization, ensuring that the model could generalize well to unseen data. The neural network model showed a marked improvement in prediction accuracy, outperforming both the BP method and traditional machine learning models. The evaluation metrics, including RMSE and Mean Absolute

Percentage Error (MAPE), confirmed the neural network's superior performance, particularly in capturing the non-linear growth patterns of the dataset. The final results were visualized through scatter plots, which illustrated a strong correlation between the predicted and actual adult heights, validating the effectiveness of the neural network approach.

Throughout the project, I collaborated closely with team members, integrating their contributions into the overall system and ensuring a cohesive and functional final product. I also documented the entire process comprehensively, providing detailed explanations and visualizations to facilitate understanding and replication of the work by others. This project allowed me to deepen my expertise in data preprocessing, exploratory data analysis, and advanced machine learning techniques, while also enhancing my skills in Python programming and experience with machine learning frameworks such as TensorFlow and Keras.