Configuration Manual

MSc Research Project

MSc in Data Analytics

Rohan Balasaheb Muradnar

Student ID:x23292113

School of Computing

National College of Ireland

Supervisor: Sallar Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name	Rohan Balasaheb Muradnar
Student ID	X23292113
Programme	Msc. In Data Analytics
Year:	2024-2025
Module:	MSc Research Project
Supervisor:	Sallar Khan
Submission Due Date:	15/09/2025
Project Title:	Configuration Manual: Bone Age Estimation Using X-ray Images with Gender-Aware Preprocessing
Word Count:	1482
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ÀLL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action

Signature : Rohan Balasaheb Muradnar

Date :15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies) **Attach a Moodle submission receipt of the online project** **submission,** to each project (including multiple copies).	□ □
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on the computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rohan Balasaheb Muradnar

Student ID: x23292113

1. Introduction

This document shows and lists down every prerequisite hardware, software, data locations, folder structure and execution commands needed to reproduce the experiments and results reported in the thesis Bone Age Estimation Using X-ray Images with Gender-Aware Pre-processing.

The pipeline is two-stage:

- Stage 1: A lightweight CNN classifies the patient's sex from the hand radiograph.
- Stage 2: Images are sent to a sex-specific InceptionV3 regressor that predicts bone-age in months.

Each step from raw RSNA/QU images to trained models and evaluation plots can be repeated by following Sections 2-8 of this manual.

Done on the full codebase, scripts to preprocess the data, train notebooks, and evaluation code can be found here: https://github.com/Rohan8586/Bone-Age-Estimation.git A README and step by step instruction on reproduction of all experiments are also available in the repository along with the cleaned metadata.

2. System Requirements

2.1. Hardware

Component	Minimum	Recommended
CPU	4 cores	8 cores+
RAM	16 GB	32 GB
GPU	(CPU works but slow)	NVIDIA GTX 1060 6 GB or better (CUDA 11)
Storage	20 GB free (images + checkpoints)	50 GB SSD

2.2. Software

Item	Version tested	Install command / link
OS	Windows 10	Ubuntu 20.04
Anaconda	2023.09-1	https://www.anaconda.com/download
Python	3.10 (conda)	Installed with Anaconda
TensorFlow /Keras	2.15.0	conda install -c conda-forge tensorflow
OpenCV-Python	4.10.0	pip install opencv-python-headless
scikit-learn	1.5.0	pip install scikit-learn
pandas	2.2.2	pip install pandas
seaborn	0.13.2	pip install seaborn
tqdm	4.66.4	pip install tqdm
Pillow	10.4.0	pip install pillow
Matplotlib	3.9.0	pip install matplotlib

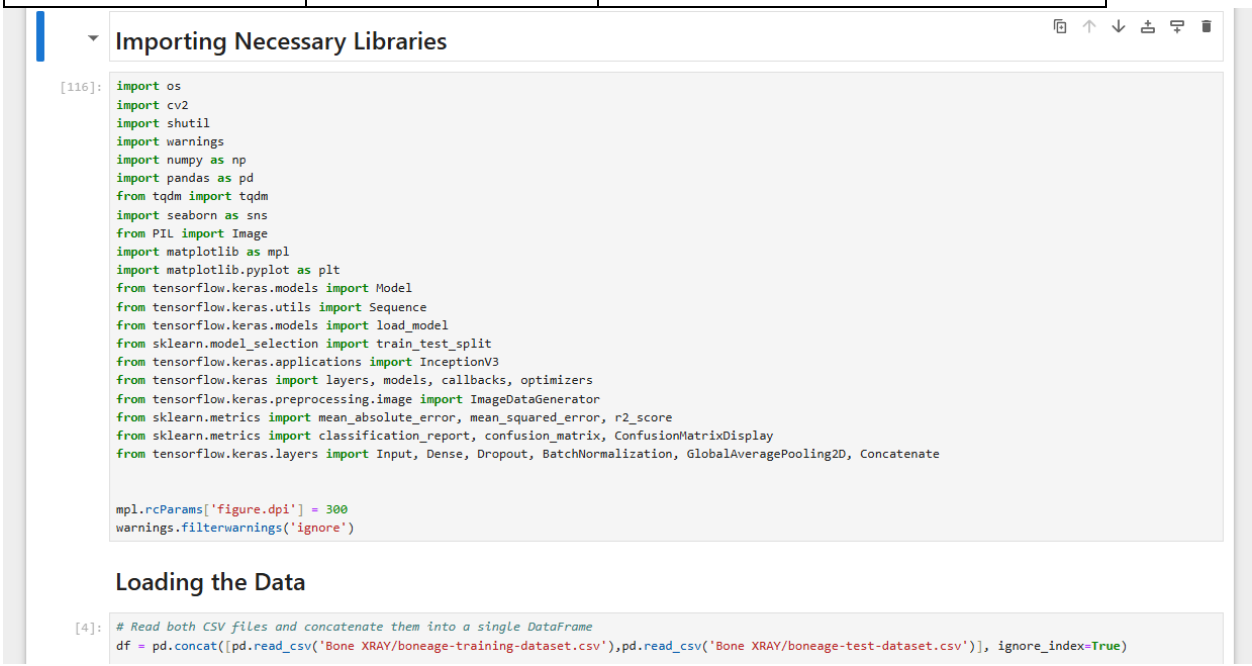

Fig-1 Python code for merging the RSNA training and test CSV files into a single DataFrame.

We start by joining the two CSV files into one table. This gives us one place to work from before cleaning anything.

3. Data Collection

Dataset	Source	Files	Size
RSNA Pediatric Bone Age	Kaggle challenge (2017)	boneage-training-dataset.csv, boneage-test-dataset.csv, corresponding *.png images	~1.4 GB
QU balanced cohort (subset used in thesis)	Provided in repo under /Bone XRAY	Already merged with RSNA metadata	

- Download the original RSNA archive (rsna-bone-age.zip) from Kaggle.
- Extract into project_root/Bone XRAY/.
- Ensure the folder names exactly match:
    - Bone XRAY/
    - ├── boneage-training-dataset/
    - ├── boneage-test-dataset/
    - ├── boneage-training-dataset.csv
    - └── boneage-test-dataset.csv

4. Data Cleaning & Pre-processing

Run Section "Dataset Preparation" in the notebook, which performs:

- CSV merge & sanity checks
- Drop nulls and duplicate rows.
- Convert id to string, map male → gender (boy|girl).
- Copy referenced PNGs into Cleaned Bone XRay/.
- Save cleaned_boneage_metadata.csv.

Loading the Data

```
[4]: # Read both CSV files and concatenate them into a single DataFrame
df = pd.concat([pd.read_csv('Bone XRAY/boneage-training-dataset.csv'),pd.read_csv('Bone XRAY/boneage-test-dataset.csv')], ignore_index=True)

print("Combined DataFrame (first 5 rows):")
display(df.head())
```

Combined DataFrame (first 5 rows):

	id	boneage	male	Case ID	Sex
0	1377.0	180.0	False	NaN	NaN
1	1378.0	12.0	False	NaN	NaN
2	1379.0	94.0	False	NaN	NaN
3	1380.0	120.0	True	NaN	NaN
4	1381.0	82.0	False	NaN	NaN

Fig-2 First five rows of the raw, concatenated metadata table (12811 records, 5 columns).

Here you can see the first few rows of that raw table. It lets us check that the columns loaded as expected.

[9]:
```
# Data types and non-null counts
print("Info:")
print(df.info())
```

```
Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12811 entries, 0 to 12810
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       12611 non-null  float64
 1   boneage  12611 non-null  float64
 2   male     12611 non-null  object
 3   Case ID  200 non-null    float64
 4   Sex      200 non-null    object
dtypes: float64(3), object(2)
memory usage: 500.6+ KB
None
```

[11]:
```
# Data types (explicit)
print("Data Types:")
print(df.dtypes)
```

```
Data Types:
id         float64
boneage    float64
male        object
Case ID    float64
Sex         object
dtype: object
```

[13]:
```
# Statistical summary
print("Statistical Summary (Numerical Columns):")
display(df.describe())
```

```
Statistical Summary (Numerical Columns):
```

	id	boneage	Case ID
count	12611.000000	12611.000000	200.000000
mean	8537.653001	127.320752	4459.500000
std	4108.763993	41.182021	57.879185
min	1377.000000	1.000000	4360.000000
25%	5074.500000	96.000000	4409.750000
50%	8565.000000	132.000000	4459.500000
75%	12091.500000	156.000000	4509.250000
max	15610.000000	228.000000	4559.000000

Fig-3 Shape, dtypes, missing-value counts and descriptive statistics of the raw DataFrame.

Now we look at the full shape, data types and missing values. These checks tell us what needs fixing next.

Data Cleaning

[22]:
```
# Drop 'Case ID' and 'Sex' columns (ignore error if already absent)
df_clean = df.drop(['Case ID', 'Sex'], axis=1, errors='ignore')

# Remove rows with any null values
df_clean = df_clean.dropna()

# Clean the 'id' column: remove decimal/trailing .0 by converting to int, then to str
df_clean['id'] = df_clean['id'].astype(int).astype(str)

# Create new 'gender' column: 'boy' where male is True, 'girl' where male is False
df_clean['gender'] = df_clean['male'].map({True: 'boy', False: 'girl', 1: 'boy', 0: 'girl', 'True': 'boy', 'False': 'girl'})

# Drop the 'male' column
df_clean = df_clean.drop('male', axis=1)

# Show new shape
print("Shape after cleaning:", df_clean.shape)

# Confirm all nulls are gone
print("\nNull Values Per Column (after cleaning):")
print(df_clean.isnull().sum())

# Show head of cleaned DataFrame
print("\nCleaned DataFrame sample:")
display(df_clean.head())
```

```
Shape after cleaning: (12611, 3)

Null Values Per Column (after cleaning):
id         0
boneage    0
gender     0
dtype: int64

Cleaned DataFrame sample:
```

	id	boneage	gender
0	1377	180.0	girl
1	1378	12.0	girl
2	1379	94.0	girl
3	1380	120.0	boy
4	1381	82.0	girl

Fig-4 Code fragment that standardises IDs and derives the categorical gender label.

This code removes unwanted columns, fixes the id, and creates a clear gender label. It prepares the data for the next steps.

[24]:
```
# --- Creating 'Cleaned Bone XRay' folder ---
cleaned_img_dir = 'Cleaned Bone XRay'
os.makedirs(cleaned_img_dir, exist_ok=True)

# --- Source directories ---
source_dirs = ['Bone XRAY/boneage-training-dataset', 'Bone XRAY/boneage-test-dataset']

# --- img_ids are already cleaned and string type ---
img_ids = df_clean['id']

missing_imgs = []

print("Copying images to Cleaned Bone XRay folder...")
for img_id in tqdm(img_ids, desc="Copying images"):
    img_id_str = str(img_id)  # Extra safety
    found = False
    for src_dir in source_dirs:
        src_img_path = os.path.join(src_dir, f"{img_id_str}.png")
        if os.path.exists(src_img_path):
            shutil.copy(src_img_path, os.path.join(cleaned_img_dir, f"{img_id_str}.png"))
            found = True
            break
    if not found:
        missing_imgs.append(img_id_str)

if missing_imgs:
    print("\nWarning: The following images were not found and could not be copied:")
    print(missing_imgs)

# --- Save the cleaned dataframe as a CSV in the cleaned images folder ---
df_clean.to_csv("cleaned_boneage_metadata.csv", index=False)
```

Fig-5 Preview of the cleaned metadata (id, boneage, gender).

The cleaned table now has three neat columns. A quick look confirms the cleaning worked.

```
Copying images to Cleaned Bone XRay folder...
Copying images: 100%|████████████████████████████████████████████████| 12611/12611 [01:20<00:00, 156.83it/s]

All cleaned images and metadata saved in: Cleaned Bone XRay
```

Fig-6 Script that copies every radiograph into a dedicated folder while logging missing files.

Next we copy every image that matches the cleaned IDs into a single folder. The loop also records any file that is missing.

```
All cleaned images and metadata saved in: Cleaned Bone XRay
```

Loading the Cleaned Data

```
[27]: # Load the cleaned DataFrame
df_cleaned = pd.read_csv("cleaned_boneage_metadata.csv")

# Display the first 5 rows
print("Loaded Cleaned DataFrame (first 5 rows):")
display(df_cleaned.head())
```

```
Loaded Cleaned DataFrame (first 5 rows):
```

	id	boneage	gender
0	1377	180.0	girl
1	1378	12.0	girl
2	1379	94.0	girl
3	1380	120.0	boy
4	1381	82.0	girl

Fig-7 Terminal confirmation that 12,611 images were successfully migrated to Cleaned Bone XRay.

The progress bar shows every file was copied. With images in place, we can move on to explore the data.

5. Exploratory Analysis

The notebook generates descriptive plots:

- Gender distribution bar-chart
- Bone-age histogram & violin plots
- Age-group × gender heat-map
- Random sample grid (Fig. 10)

These steps are optional for pure replication but help verify the dataset integrity.

Exploratory Data Analysis

```
[29]: # 1. Gender distribution (updated for new 'gender' column)
sns.countplot(x='gender', data=df_clean)
plt.title("Gender Distribution")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.show()
```

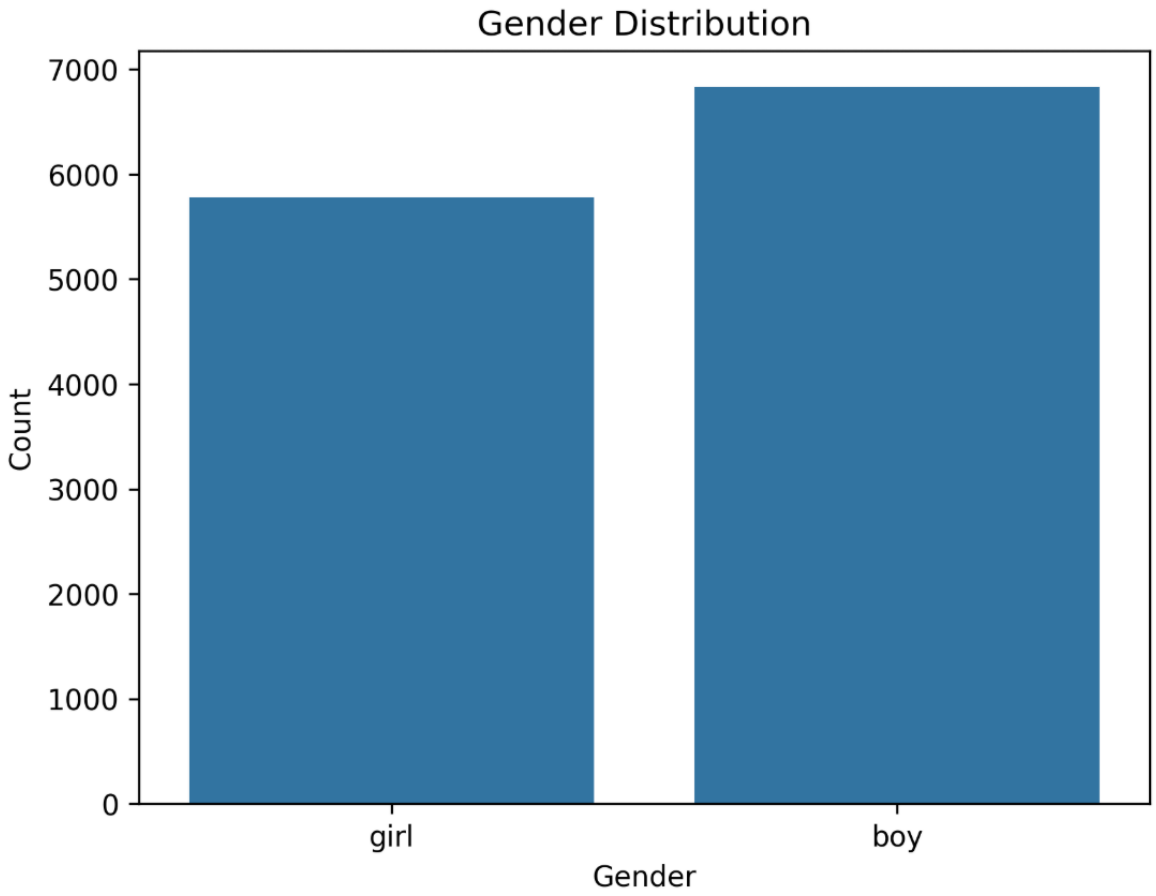

Fig-8 Bar chart showing an almost perfectly balanced number of boys and girls.

First, we count how many boys and girls are in the set. A balanced count is good for training.

```
[31]: # 2. Boneage distribution
sns.histplot(df_clean['boneage'], bins=30, kde=True)
plt.title("Boneage Distribution")
plt.xlabel("Bone Age (Months)")
plt.ylabel("Count")
plt.show()
```

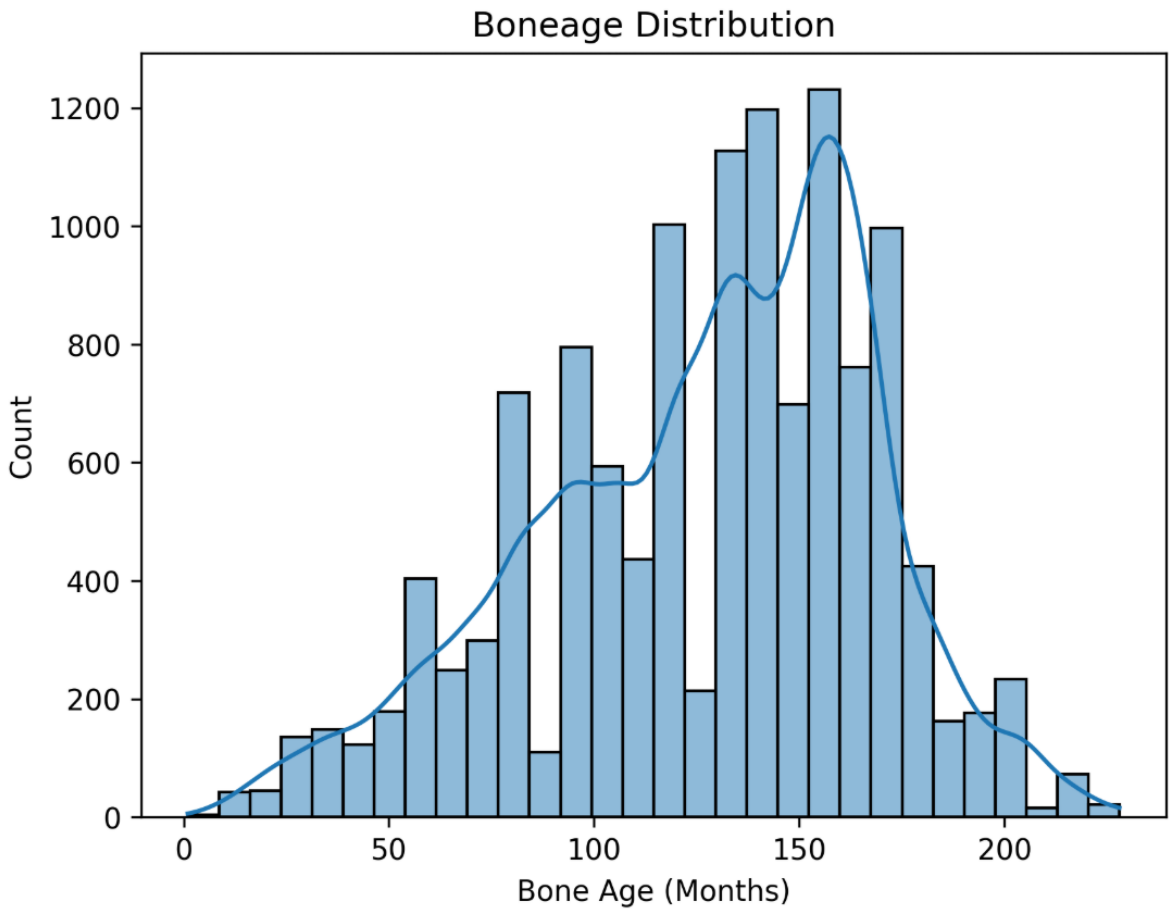

Fig-9 Bone-age histogram (1–228 months) with kernel density overlay.

Then we plot how bone age is spread across all children. This shows the range and most common ages.

```
[33]: # 3. Boneage by gender
sns.boxplot(x='gender', y='boneage', data=df_clean)
plt.title("Boneage by Gender")
plt.xlabel("Gender")
plt.ylabel("Bone Age (Months)")
plt.show()
```

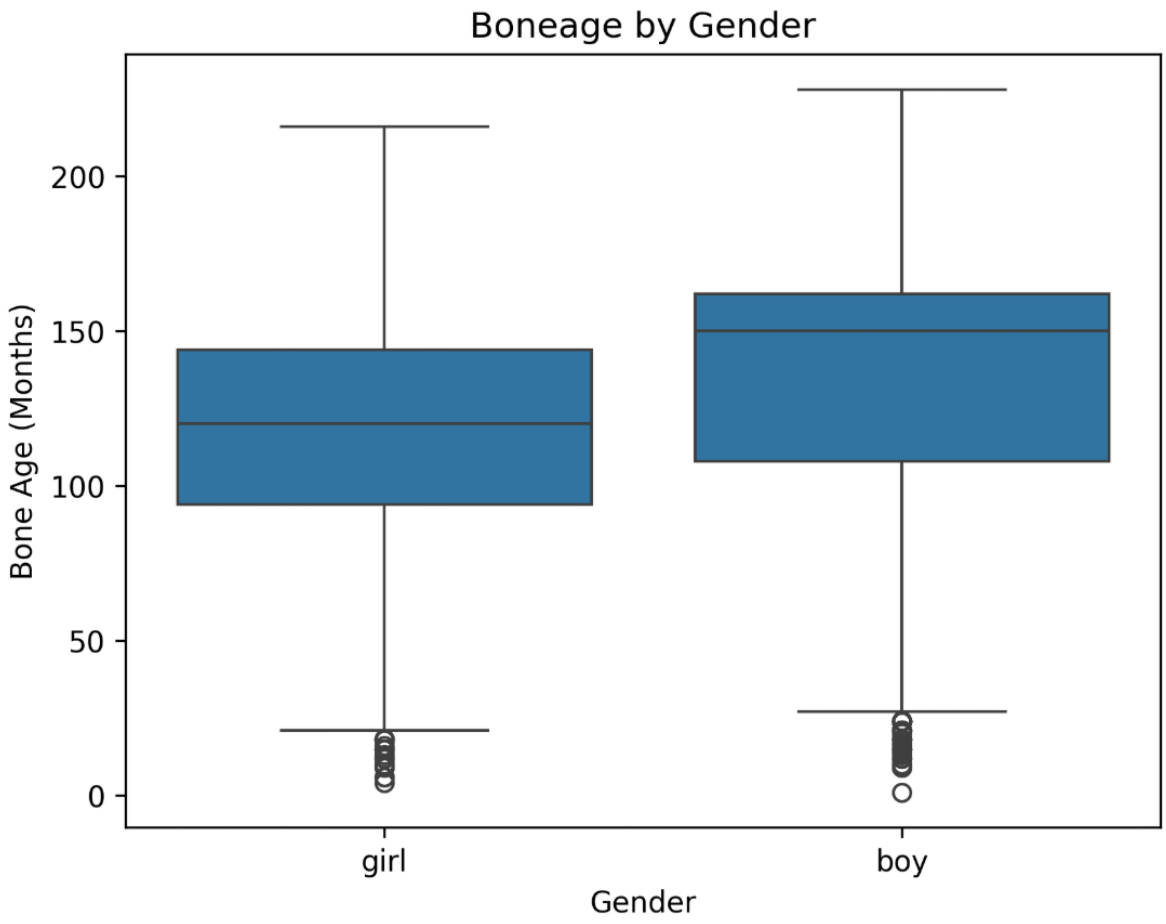

Fig-10 Box-plot highlighting median age differences and outliers between the sexes.

We split that age spread by gender to spot any clear differences. The box-plot makes those gaps easy to see.

[37]:
```
# 5. Age Group Analysis (in months)
# Define bins and labels for months
bins = [0, 84, 144, 180, 228]
labels = ['Early childhood (0-7y)', 'Late childhood (8-12y)', 'Puberty (13-15y)', 'Late teens (16-19y)']

# Create age group Series (does NOT modify df_clean)
age_group = pd.cut(df_clean['boneage'], bins=bins, labels=labels, right=True, include_lowest=True)

# Plot countplot by age group
plt.figure(figsize=(8,5))
sns.countplot(x=age_group, order=labels)
plt.title('Boneage Distribution by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Number of Cases')
plt.tight_layout()
plt.show()
```

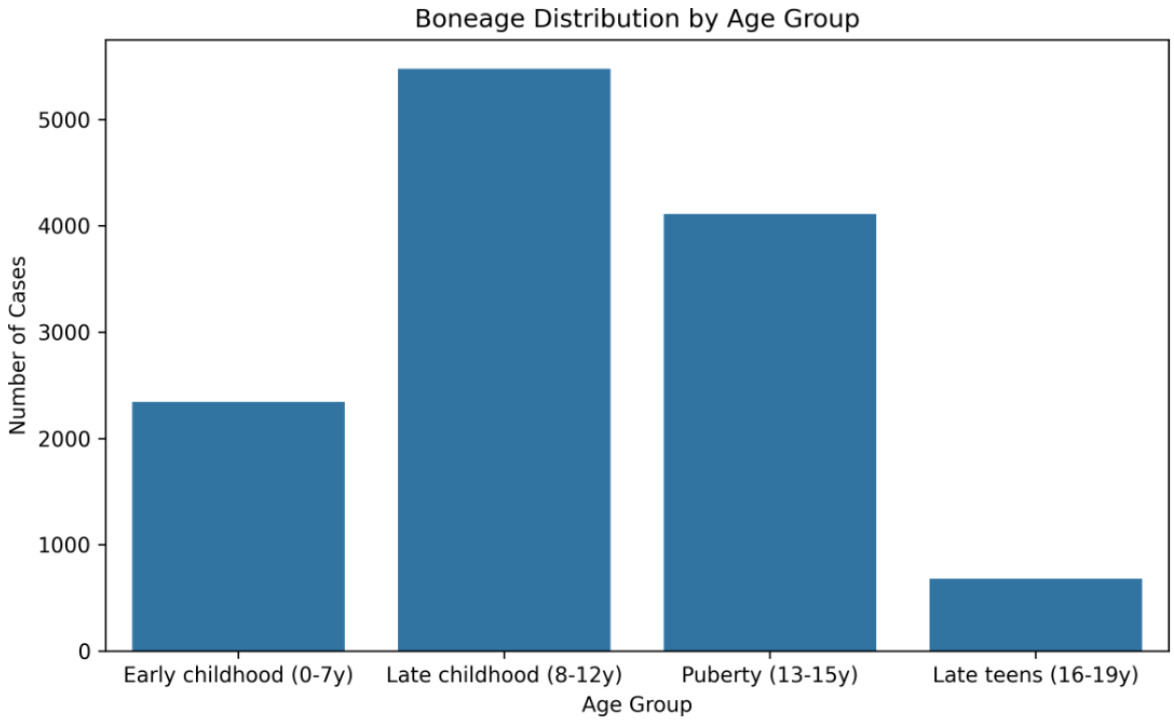

Fig-11 Distribution of cases across the four developmental age-brackets.

Ages are grouped into four life stages. The bar chart shows how many cases fall into each stage.

[39]:
```
# 6. Gender Distribution Within Age Groups

# Define bins and labels for months
bins = [0, 84, 144, 180, 228]
labels = ['Early childhood (0-7y)', 'Late childhood (8-12y)', 'Puberty (13-15y)', 'Late teens (16-19y)']

# Compute age groups on-the-fly
age_group = pd.cut(df_clean['boneage'], bins=bins, labels=labels, right=True, include_lowest=True)

plt.figure(figsize=(10,6))
sns.countplot(x=age_group, hue=df_clean['gender'], order=labels)
plt.title('Boneage Distribution by Age Group and Gender')
plt.xlabel('Age Group')
plt.ylabel('Number of Cases')
plt.legend(title='Gender')
plt.tight_layout()
plt.show()
```

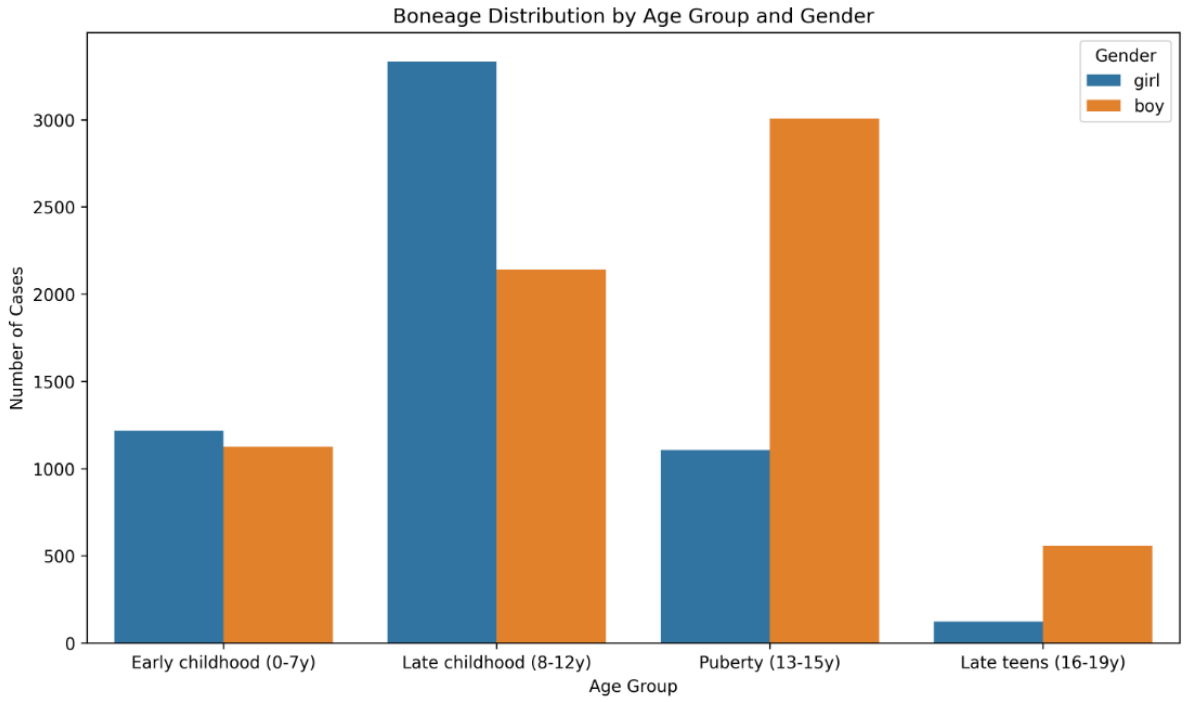

Fig-12 Joint distribution of age-group and gender useful for later stratified testing.

We add colour by gender to the same age-group plot. This checks that each stage has both boys and girls.

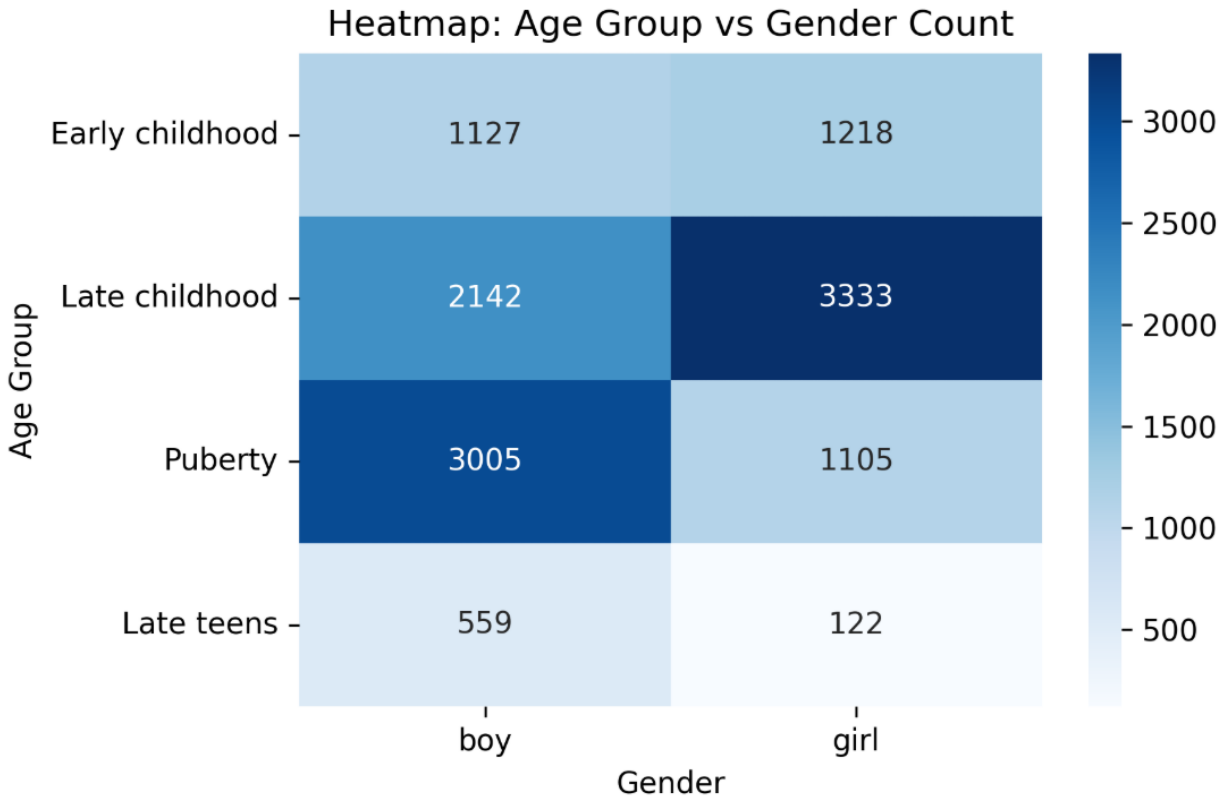

Fig-13 Heat-map of counts per (age-group × gender) cell.

The heat-map puts the same counts into a grid. It helps us spot any empty or very small groups.

6. Image Pipeline

preprocess_pil_image()

- CLAHE (clipLimit 2.0, tileGrid 8×8)
- Otsu threshold + largest-contour crop
- 3 % margin (boys) / 2 % margin (girls)
- Resize → 224 for Stage 1, 299 for Stage 2
- Normalise → [0,1] and replicate grey channel to RGB

Two custom Sequence generators serve the models:

Class	Target	Output
CustomHandXraySequence	Stage 1 gender CNN	(batch,224,224,3) , gender_num
XrayGenderSeq	Stage 2 regressors	([image, gender_num], boneage)

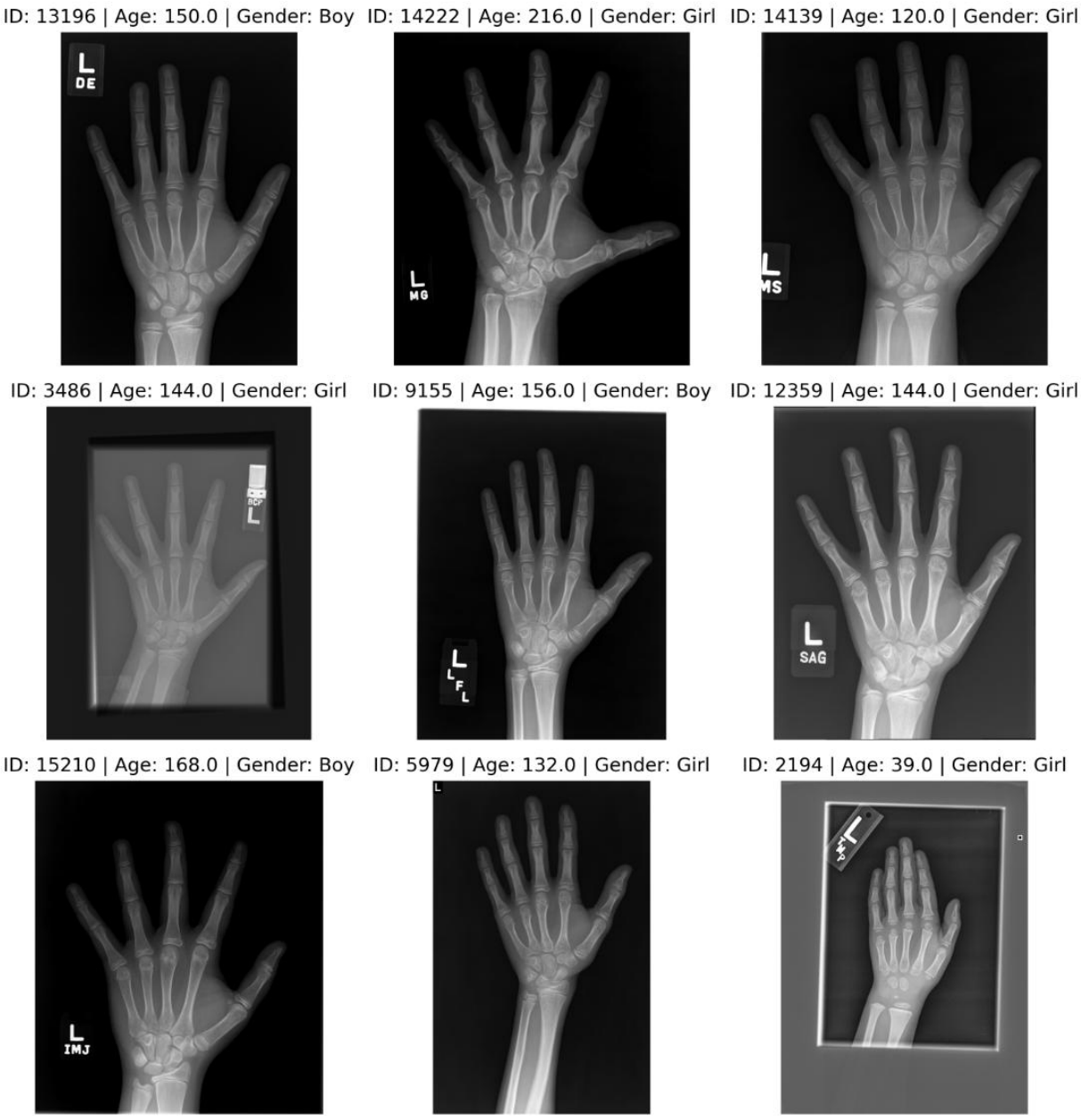

Fig-14 Random raw radiographs with ground-truth age and gender labels.

Here are nine raw X-rays straight from the dataset. Seeing them helps us understand what the model will view.

Data Preprocessing

[29]:
```python
def preprocess_pil_image(pil_img, output_size=(224, 224)):
    # Convert PIL image to numpy grayscale
    img = np.array(pil_img.convert('L'))

    # CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    img_clahe = clahe.apply(img)

    # Otsu thresholding
    _, thresh = cv2.threshold(img_clahe, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    if np.mean(thresh[:10, :10]) > 127:
        thresh = 255 - thresh
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if not contours:
        hand_roi = img_clahe
    else:
        c = max(contours, key=cv2.contourArea)
        x, y, w, h = cv2.boundingRect(c)
        margin = int(0.03 * max(w, h))
        x1, y1 = max(x - margin, 0), max(y - margin, 0)
        x2, y2 = min(x + w + margin, img_clahe.shape[1]), min(y + h + margin, img_clahe.shape[0])
        hand_roi = img_clahe[y1:y2, x1:x2]

    # Resize and normalize
    hand_resized = cv2.resize(hand_roi, output_size, interpolation=cv2.INTER_AREA)
    hand_norm = hand_resized.astype(np.float32) / 255.0
    # Convert single channel to 3-channel image
    img_rgb = np.stack([hand_norm]*3, axis=-1)
    return img_rgb
```

Fig-15 Custom function performing CLAHE, Otsu threshold, margin crop and resize.

This function cleans each image: better contrast, crops the hand, and resizes it. Clean images train better models.

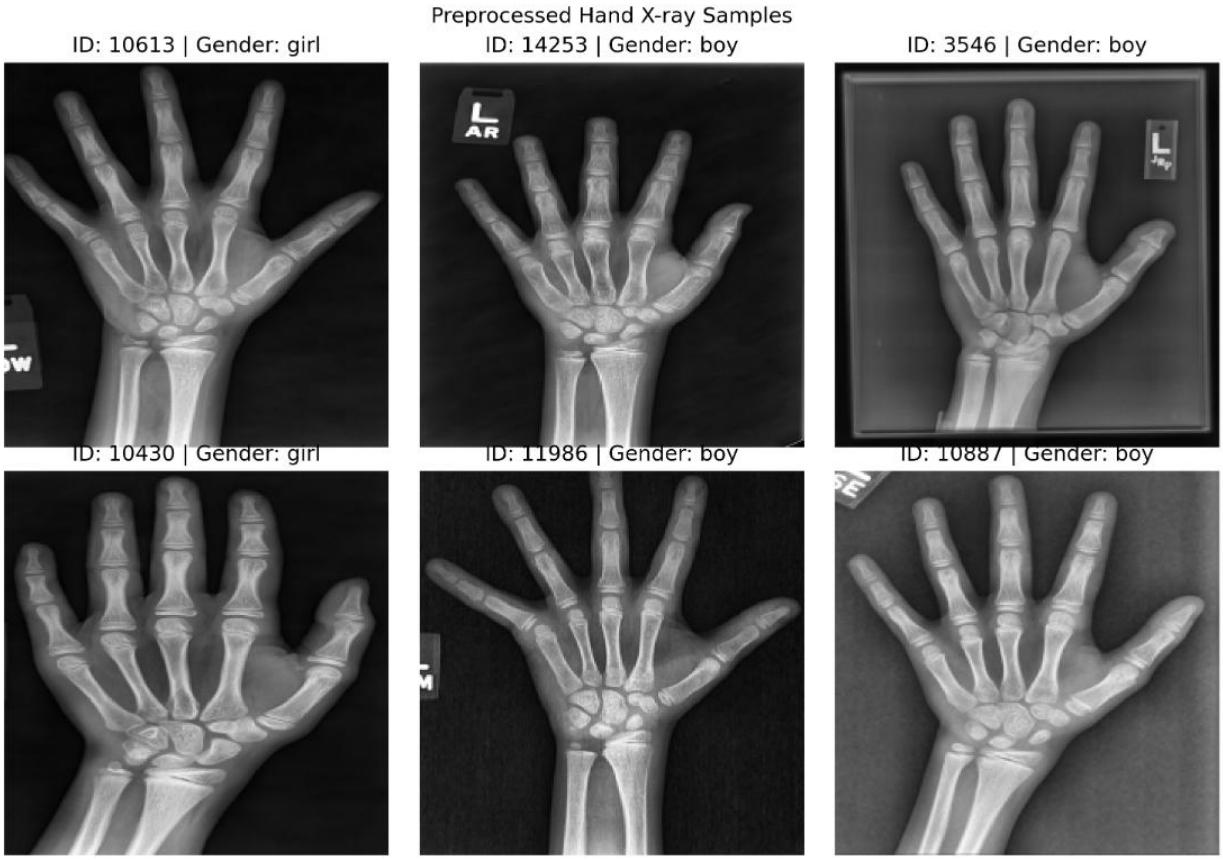

Fig-16 Examples of contrast-enhanced, cropped and normalised inputs (224 × 224 px).

These six pictures show how the images look after cleaning. They are now all the same size and style.

```
[44]: class CustomHandXraySequence(Sequence):
          def __init__(self, df, batch_size=32, img_size=(224,224), shuffle=True):
              self.df = df.reset_index(drop=True)
              self.batch_size = batch_size
              self.img_size = img_size
              self.shuffle = shuffle
              self.indexes = np.arange(len(df))
              self.on_epoch_end()

          def __len__(self):
              return int(np.ceil(len(self.df) / self.batch_size))

          def on_epoch_end(self):
              if self.shuffle:
                  np.random.shuffle(self.indexes)

          def __getitem__(self, idx):
              inds = self.indexes[idx * self.batch_size : (idx + 1) * self.batch_size]
              X = []
              y = []
              for i in inds:
                  row = self.df.iloc[i]
                  pil_img = Image.open(os.path.join('Cleaned Bone XRay', f"{row['id']}.png"))
                  img_arr = preprocess_pil_image(pil_img, output_size=self.img_size)
                  X.append(img_arr)
                  y.append(row['gender_num'])
              return np.stack(X), np.array(y)
```

Fig-17 Data-generator class that streams mini-batches for the sex-classification CNN.

The generator code feeds images to the model in batches. It also keeps labels lined up with the right images.

7. Model Training

7.1. Stage 1 – Gender Classifier

Setting	Value
Architecture	4 × Conv-BN-ReLU-Pool blocks → Flatten → Dense 128 → Sigmoid
Params	≈ 6.8 M
Loss / Opt	Binary-Cross-Entropy / Adam 1e-3
Callbacks	EarlyStopping (patience 6) ReduceLROnPlateau (×0.3)

Defining Model

[49]:
```
IMG_SIZE = (224, 224, 3)

# Define custom CNN
model_gender = models.Sequential([
    layers.Input(shape=IMG_SIZE),

    layers.Conv2D(32, (3, 3), padding='same', kernel_initializer='he_normal'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),

    layers.Conv2D(64, (3, 3), padding='same', kernel_initializer='he_normal'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),

    layers.Conv2D(128, (3, 3), padding='same', kernel_initializer='he_normal'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.3),

    layers.Conv2D(256, (3, 3), padding='same', kernel_initializer='he_normal'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.4),

    layers.Flatten(),
    layers.Dense(128, activation='relu', kernel_initializer='he_normal'),
    layers.BatchNormalization(),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')  # Binary gender output
])

# Compile model
model_gender.compile(
    optimizer=optimizers.Adam(learning_rate=1e-3),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Callbacks
early_stop = callbacks.EarlyStopping(
    monitor='val_loss', patience=6, restore_best_weights=True, verbose=1
)
reduce_lr = callbacks.ReduceLROnPlateau(
    monitor='val_loss', factor=0.3, patience=3, min_lr=1e-6, verbose=1
)

# Model summary
model_gender.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 224, 224, 32)      896

 batch_normalization_5 (Batc  (None, 224, 224, 32)     128
 hNormalization)
```

Fig-18 Architecture of the lightweight four-block CNN used for automatic sex prediction.

This is the small CNN that guesses the child's sex. It has four convolution blocks followed by dense layers.

```python
plt.figure(figsize=(14, 5))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Model Accuracy')

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Model Loss')

plt.tight_layout()
plt.show()
```

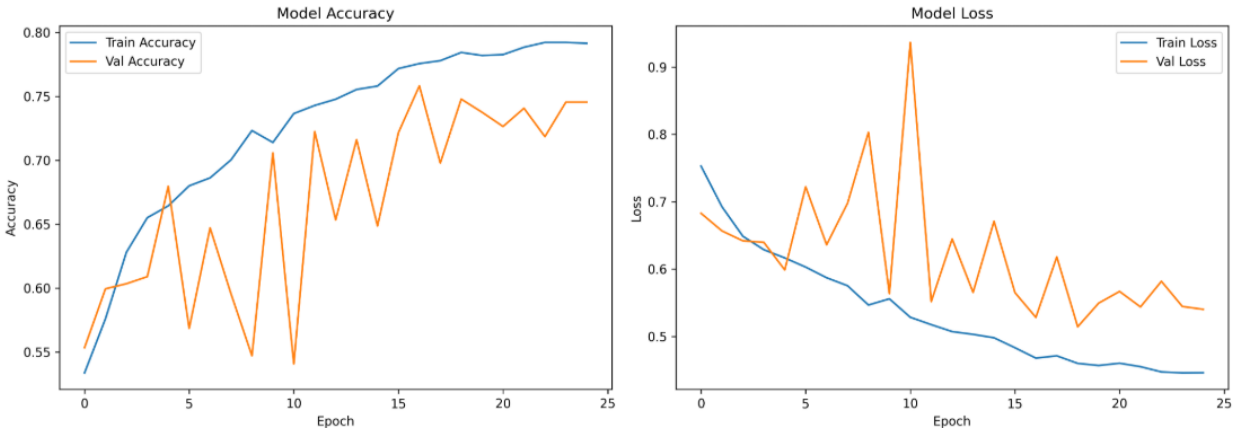

Fig-19 Convergence curves for the sex-classifier (early-stopped after ≈ 20 epochs).

Training and validation curves let us see if learning is steady. They show the model stops improving after about 20 epochs.

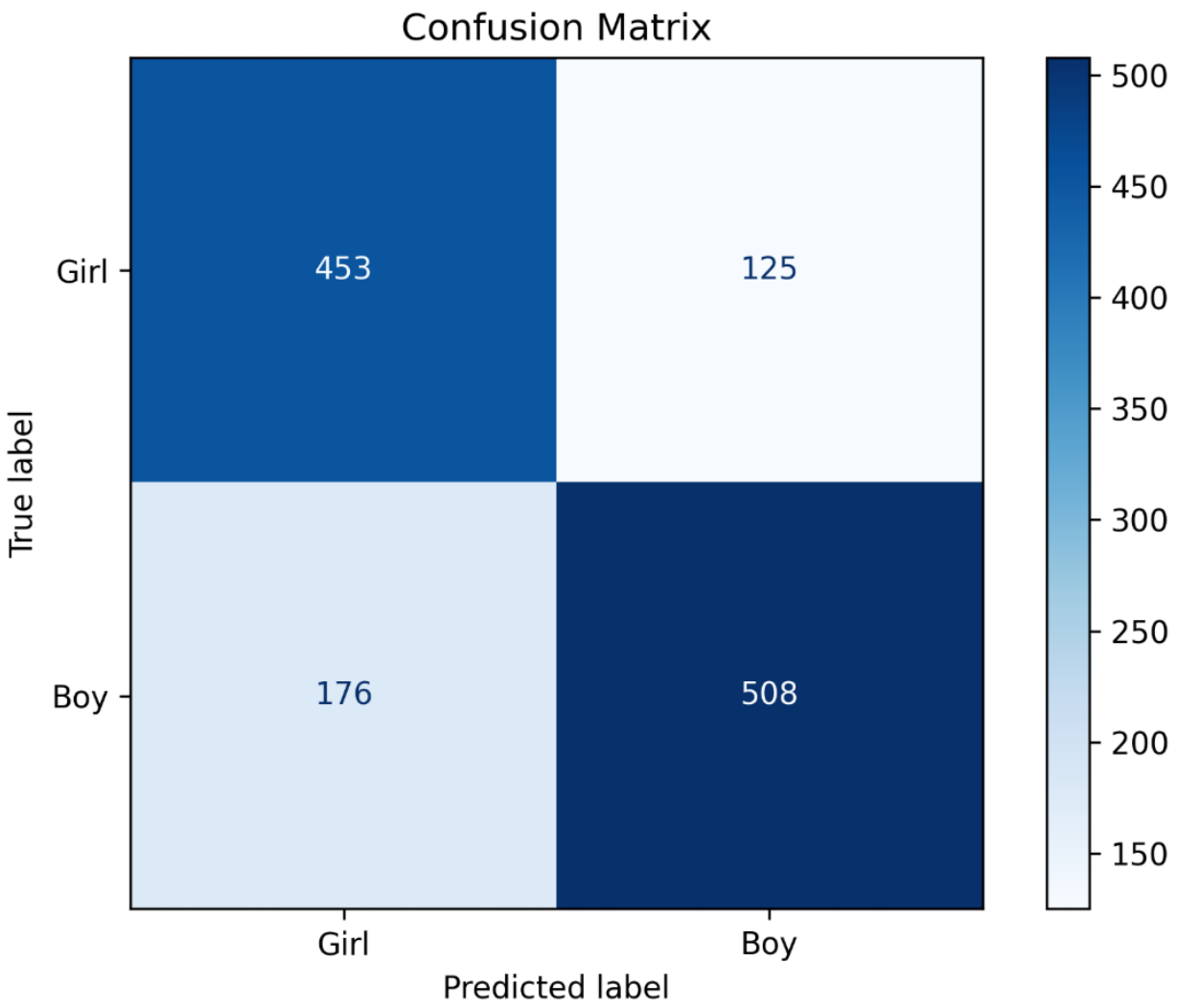

Fig-20 Confusion matrix on the held-out test split (overall accuracy ≈ 76 %).

The confusion matrix tells us how often the model is right or wrong for each sex. It sums up the classifier's real performance.

7.2. Stage 2 – Bone-Age Regressor

Setting	Value
Backbone	InceptionV3 (ImageNet) frozen warm-up, then full fine-tune
Extra heads	GlobalAvgPool → Dense 256 → concat gender (32-unit) → Dense 128 → output
Loss	MAE
Opt / LR	Adam 1e-4 with ReduceLROnPlateau
Metrics	MAE, MSE (R^2 computed manually)

After 10 epochs as in thesis, Test MAE ≈ 50.6 months (baseline run).

Weights saved as bone_age_estimator.keras.

Saving the model

[61]:
```
# Save entire model (architecture, weights, optimizer state, etc.)
model_gender.save('gender_classifier_model.keras')
```

Age Estimation Model

Data Processing

[70]:
```
# Stratify by gender to preserve distribution
train_df, temp_df = train_test_split(
    df_clean, test_size=0.2, stratify=df_clean['gender_num'], random_state=42)
val_df, test_df = train_test_split(
    temp_df, test_size=0.5, stratify=temp_df['gender_num'], random_state=42)
```

[74]:
```
class XrayGenderSeq(Sequence):
    def __init__(self, df, batch_size=32, img_size=(299, 299), shuffle=True):
        self.df = df.reset_index(drop=True)
        self.batch_size = batch_size
        self.img_size = img_size
        self.shuffle = shuffle
        self.indexes = np.arange(len(self.df))
        self.on_epoch_end()

    def __len__(self):
        return int(np.ceil(len(self.df) / self.batch_size))

    def on_epoch_end(self):
        if self.shuffle:
            np.random.shuffle(self.indexes)

    def __getitem__(self, idx):
        inds = self.indexes[idx*self.batch_size:(idx+1)*self.batch_size]
        X_img = []
        X_gender = []
        y = []
        for i in inds:
            row = self.df.iloc[i]
            pil_img = Image.open(os.path.join('Cleaned Bone XRay', f"{row['id']}.png"))
            img_arr = preprocess_pil_image(pil_img, output_size=self.img_size)
            X_img.append(img_arr)
            X_gender.append(row['gender_num'])
            y.append(row['boneage'])
        return [np.stack(X_img), np.array(X_gender).reshape(-1,1)], np.array(y).astype(np.float32)
```

[76]:
```
train_seq = XrayGenderSeq(train_df, batch_size=32, img_size=(299,299), shuffle=True)
val_seq   = XrayGenderSeq(val_df,   batch_size=32, img_size=(299,299), shuffle=False)
test_seq  = XrayGenderSeq(test_df,  batch_size=32, img_size=(299,299), shuffle=False)
(X_img_batch, X_gender_batch), y_batch = train_seq[0]
print("Image batch:", X_img_batch.shape)
print("Gender batch:", X_gender_batch.shape)
print("Bone age batch:", y_batch.shape)
```

```
Image batch: (32, 299, 299, 3)
Gender batch: (32, 1)
Bone age batch: (32,)
```

Fig-21 Gender-aware InceptionV3 regression head with late fusion of the sex flag.

We now build the age-prediction model. It uses InceptionV3 plus the sex flag as extra input.

```python
plt.figure(figsize=(14, 5))

# Plot MAE
plt.subplot(1, 2, 1)
plt.plot(history_age.history['mae'], label='Train MAE')
plt.plot(history_age.history['val_mae'], label='Val MAE')
plt.xlabel('Epoch')
plt.ylabel('MAE')
plt.title('Mean Absolute Error (MAE) Over Epochs')
plt.legend()

# Plot MSE
plt.subplot(1, 2, 2)
plt.plot(history_age.history['mse'], label='Train MSE')
plt.plot(history_age.history['val_mse'], label='Val MSE')
plt.xlabel('Epoch')
plt.ylabel('MSE')
plt.title('Mean Squared Error (MSE) Over Epochs')
plt.legend()

plt.tight_layout()
plt.show()
```

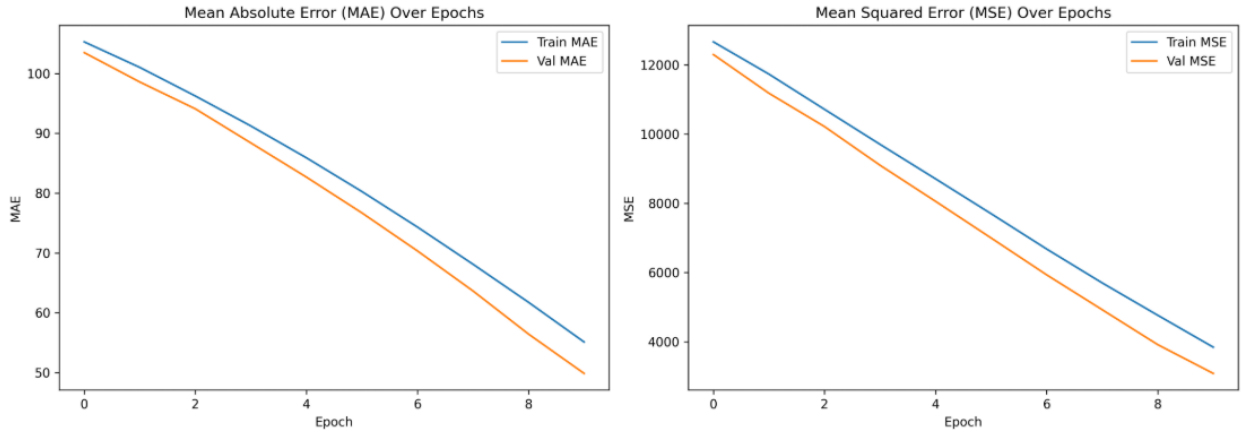

Fig-22 MAE and MSE learning curves for the bone-age regressor (10 epochs shown).

These curves show how the age model's error drops during training. Lower MAE and MSE mean more accurate ages.

8. Evaluation & Visualisation

- classification_report() + confusion matrix for Stage 1
- Per-image MAE/MSE/R^2 on test split for Stage 2
- Percentage within ±6 / ±12 months
- Calibration and Bland-Altman plots (notebook cells labelled Calibration).
- Grad-CAM visualisations optional (commented utilities included).

To re-display five random test predictions with overlays, run the final cell "Load models … Model Predictions vs Ground Truth".

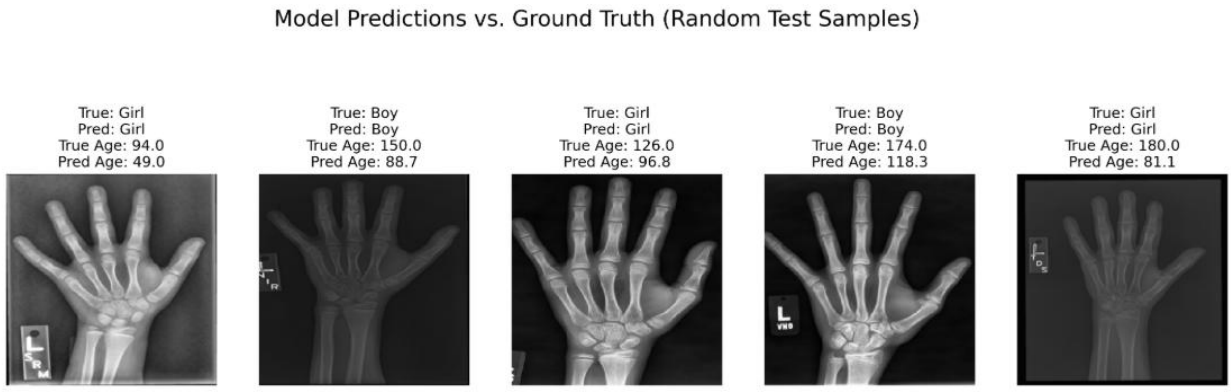

Fig-23 End-to-end demo: true vs predicted sex and bone-age on five unseen radiographs.

Finally, we test the full pipeline on five new images. The panel shows true labels beside the model's guesses.

9. References

Abadi, M. et al. (2016) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available at: https://www.tensorflow.org

Bradski, G. (2000) 'The OpenCV Library', Dr. Dobb's Journal of Software Tools. Available at: https://opencv.org

Chollet, F. (2015) Keras: Deep Learning for Humans. GitHub repository, https://github.com/keras-team/keras

Halabi, S.S. et al. (2018) 'The RSNA Pediatric Bone Age Machine Learning Challenge', Radiology, 290(2), pp. 498–503. https://doi.org/10.1148/radiol.2018180736

Harris, C.R. et al. (2020) 'Array Programming with NumPy', Nature, 585, pp. 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J.D. (2007) 'Matplotlib: A 2D Graphics Environment', Computing in Science & Engineering, 9(3), pp. 90–95. https://doi.org/10.1109/MCSE.2007.55

Kaggle (2017) RSNA Bone Age Dataset. Available at: https://www.kaggle.com/competitions/rsna-bone-age