Bone Fracture Analysis & Classification Using Deep Learning Models

Kunta Sreeja[1], Bandari Sharvan[2], Panthulu Ravi Raja[3], Gundrala Mohan Aditya[4], Sirugumalle Anusha[5]

[1,2,3,4]Department of Information Technology,
J.B. Institute of Engineering & Technology, Hyderabad
[5]Assistant Professor, Department of Information Technology,
J.B. Institute of Engineering & Technology, Hyderabad

***Abstract*—Fracture diagnosis from musculoskeletal radiographs demands both speed and precision, yet manual interpretation remains prone to inter-observer variability and fatigue-driven error. This paper presents BoneAI, a web-deployed two-stage deep learning framework that automates bone fracture detection in X-ray images of the elbow, hand, and shoulder. The first stage employs a fine-tuned ResNet50 convolutional neural network to identify the anatomical body part from the radiograph; the second stage applies a separately trained ResNet50 classifier to determine whether a fracture is present. Both models are pre-trained on ImageNet and fine-tuned on a balanced musculoskeletal dataset, with the last twenty ResNet50 layers unfrozen during fine-tuning at a reduced learning rate of 1×10^{-5}. To provide transparency in predictions, Gradient-weighted Class Activation Mapping (Grad-CAM) is integrated into the inference pipeline, producing colour-coded heat maps that visually localise the regions of highest model attention on the original radiograph. The system additionally supports DICOM file ingestion through automatic pixel normalisation and format conversion, making it compatible with standard hospital imaging workflows. A Django-based web application exposes the full pipeline through a browser interface and delivers downloadable PDF medical reports containing the original image, the Grad-CAM overlay, AI observation text, and clinical recommendations. Experiments on the musculoskeletal dataset demonstrate reliable classification performance across the three supported body parts. The proposed framework reduces reporting time while improving interpretability, and offers a practical screening aid for radiologists and orthopaedic clinicians.**

Keywords: bone fracture detection, deep learning, ResNet50, transfer learning, Grad-CAM, musculoskeletal radiograph, DICOM, explainable AI, web-based clinical tool, two-stage classification.

I. INTRODUCTION

Bone fractures represent one of the most common musculoskeletal injuries encountered in emergency departments worldwide. Early and accurate detection is essential for timely clinical intervention, because delayed or missed diagnoses can lead to complications such as malunion, avascular necrosis, and long-term functional impairment [2]. Plain radiography remains the first-line imaging modality for suspected fractures owing to its cost-effectiveness and widespread availability. However, interpreting radiographs demands years of specialist training, and the growing demand for diagnostic imaging has placed considerable pressure on radiology departments, creating a need for computer-aided diagnostic (CAD) tools that can assist in triage and preliminary screening.

Early CAD systems for fracture detection relied on hand-crafted image features such as edge detection, gradient orientation histograms, and morphological analysis. While these approaches offered some benefit, they generalised poorly across different imaging conditions, patient populations, and anatomical sites. The emergence of deep convolutional neural networks (CNNs) fundamentally changed this landscape. Since AlexNet demonstrated the power of learned hierarchical features on ImageNet [8], deeper and more sophisticated architectures such as VGGNet [7], GoogLeNet, and ResNet [4] have pushed classification accuracy well beyond previously achievable benchmarks.

ResNet50, introduced by He *et al.* in 2016, addressed the long-standing problem of vanishing gradients in very deep networks through identity shortcut connections, also known as residual connections. This architectural innovation allows gradients to propagate cleanly through fifty layers, enabling the network to learn both fine-grained local textures and high-level semantic representations simultaneously. When applied through transfer learning — where ImageNet-trained weights are used as a starting point for domain-specific fine-tuning — ResNet50 has shown strong performance on medical imaging tasks even with comparatively small labelled datasets [4].

Despite the progress in classification accuracy, a recurring criticism of deep learning models in clinical settings is their "black box" nature. Clinicians require not only a prediction but also some indication of the spatial basis for that prediction. Gradient-weighted Class Activation Mapping (Grad-CAM), proposed by Selvaraju *et al.* [3], addresses this by computing the gradient of the class score with respect to the final convolutional feature maps, producing a localisation heat map that highlights the image regions most influential for the network's decision. Grad-CAM has since become a standard interpretability technique in medical imaging research.

Parallel to these modelling advances, the adoption of DICOM (Digital Imaging and Communications in Medicine) as the universal standard for medical image storage means that any clinically viable tool must handle DICOM inputs natively. Libraries such as pydicom [9] enable Python applications to read DICOM files, extract pixel arrays, and apply appropriate

normalisation before feeding images to deep learning models.

This paper addresses all the above considerations in a single integrated system named BoneAI. The specific contributions of this work are as follows.

1) A two-stage inference pipeline that first classifies the anatomical body part (elbow, hand, or shoulder) and then determines fracture status, enabling the fracture classifier to specialise on a known anatomical region.
2) Architecture-aware Grad-CAM implementation that correctly traverses the nested ResNet50 sub-model used in training, producing accurate colour-coded overlays on each predicted image.
3) Full DICOM support through automatic pixel array extraction, windowing-aware normalisation, and conversion to the RGB format expected by the model.
4) A pixel-level saturation validator that rejects non-X-ray uploads (colour photographs, screenshots) before expensive model inference is invoked, combined with a model-confidence guard that enforces a minimum certainty threshold.
5) A Django web application with role-based user management, prediction history, analytics dashboards, and downloadable PDF medical reports that embed the Grad-CAM overlay alongside structured AI observation text.

The remainder of the paper is organised as follows. Section II reviews related work. Section III describes existing systems and their limitations. Section IV presents the proposed methodology at a high level. Section V details the dataset, model architecture, training procedure, and Grad-CAM formulation. Section VI covers the system implementation. Section VII presents results and discussion, and Section VIII concludes with directions for future work.

II. Literature Survey

Research in automated fracture detection has progressed steadily from classical image-processing pipelines to modern end-to-end deep learning systems, and the following review traces that trajectory with emphasis on the methodological choices most relevant to the present work.

Rajpurkar *et al.* released the MURA (Musculoskeletal Radiographs) dataset in 2018, comprising over forty thousand musculoskeletal radiographs spanning seven body parts, each labelled normal or abnormal by board-certified radiologists [2]. The authors trained a 169-layer DenseNet on this dataset and reported performance comparable to that of radiologists on several body parts. Their study established that large-scale, body-part-specific datasets are essential for training reliable fracture detection models and that the anatomical category of a radiograph carries significant predictive value — motivating the two-stage architecture adopted in the present work.

He *et al.* proposed the Residual Network (ResNet) family in 2016, demonstrating that identity shortcut connections permit the training of networks up to 152 layers without degradation [4]. The fifty-layer variant (ResNet50) strikes an effective balance between representational capacity and computational cost, and has consequently become the most widely used backbone in transfer learning for medical imaging. The mathematical formulation of the residual block and its role in gradient propagation are discussed in detail in Section V.

Simonyan and Zisserman introduced VGGNet in 2014, demonstrating the value of uniform 3×3 convolution kernels arranged in deep sequences [7]. Although VGG16 and VGG19 remain useful baselines, their parameter counts are significantly larger than ResNet50 for comparable depth, making them less practical for fine-tuning on medical datasets with limited computational resources.

Transfer learning in medical imaging was systematically studied by Shin *et al.* [5], who examined how the depth of fine-tuning affects performance on radiology tasks. They found that fine-tuning deeper portions of an ImageNet-pretrained network generally improves results on medical images, particularly when the source and target domains differ in texture statistics. This finding directly informed the fine-tuning strategy in BoneAI, where the last twenty layers of ResNet50 are unfrozen in a second training phase at a reduced learning rate.

Selvaraju *et al.* introduced Grad-CAM in 2017 as a generalisation of Class Activation Mapping (CAM) that does not require architectural modifications or re-training [3]. Grad-CAM uses the gradient of the class score with respect to the activations of the last convolutional layer to weight each feature map, then applies a ReLU to retain only those spatial positions that positively influence the prediction. The resulting heat map can be upsampled to the input resolution and superimposed on the original image. In medical imaging, Grad-CAM overlays have been validated by radiologists as accurately indicating pathology locations in chest X-rays, brain MRI, and retinal fundus images, lending credibility to their use in the present fracture detection context.

The problem of model interpretability in clinical AI has been discussed broadly in the literature. Tjoa and Guan [1] surveyed explainable AI methods in healthcare and concluded that gradient-based attribution methods such as Grad-CAM provide the best combination of computational efficiency and spatial fidelity for image classification tasks. Their survey underlines the clinical necessity of producing human-readable justifications alongside automated predictions, which is a core design goal of BoneAI.

DICOM handling in Python-based medical applications has been facilitated by the pydicom library [9], which provides a straightforward API for reading DICOM metadata and pixel arrays. Pixel values in DICOM files are encoded in modality-specific Hounsfield units or raw detector counts and require windowing or linear normalisation before being processed by networks trained on natural images. The present system applies min-max normalisation to map pixel values to the [0, 255] range prior to conversion to RGB, following practices established in the clinical imaging literature.

Web-based deployment of clinical AI tools has been explored in several recent studies. Django, a high-level Python web framework following the model-view-template pattern, has been used to host medical AI services due to its ma-

ture authentication, ORM, and file-handling subsystems. The combination of Django with a TensorFlow backend allows inference to be decoupled from the user interface through lazy model loading, so the computational overhead of loading ResNet50 weights is incurred only once per server session [6].

Taken together, the literature highlights three gaps that the present work addresses: (i) the absence of a body-part-aware two-stage pipeline for musculoskeletal fracture detection in web-deployed systems; (ii) the lack of architecture-aware Grad-CAM implementations that correctly handle nested sub-model structures produced by the standard Keras functional API; and (iii) the need for integrated DICOM ingestion with automatic pixel normalisation in browser-accessible fracture screening tools.

III. Existing System

A. Manual Radiograph Interpretation

The conventional approach to fracture diagnosis relies entirely on manual inspection of radiographs by radiologists and orthopaedic clinicians. A trained specialist visually examines cortical discontinuities, trabecular disruption, periosteal reactions, and soft-tissue changes to reach a diagnostic conclusion. While expert radiologists achieve high sensitivity on obvious fractures, studies have documented notable inter-observer variability, particularly for non-displaced, stress, and occult fractures [2]. Furthermore, radiology departments in many healthcare systems are under increasing workload pressure, which contributes to fatigue-driven reporting errors and extended turnaround times in high-volume emergency settings.

B. Classical Computer-Aided Detection Systems

Early computer-aided detection (CAD) systems for fracture detection were built on hand-engineered image features. Common pipelines involved Canny or Sobel edge detection to extract bone boundaries, Hough transforms to locate cortical outlines, and morphological operations to identify discontinuities consistent with fractures. Although these methods provided interpretable outputs, they were highly sensitive to acquisition parameters such as exposure, tube voltage, and patient positioning. Small changes in imaging protocol could invalidate the tuned thresholds, leading to unacceptable false-positive rates in practice. More importantly, classical CAD systems were designed for a single anatomical site and could not generalise across body parts without complete re-engineering.

C. Single-Stage CNN Classifiers

The success of deep CNNs on natural image benchmarks motivated their application to musculoskeletal radiographs as single-stage classifiers. In this paradigm, a single network is trained end-to-end to predict fracture status directly from an image, regardless of the body part it depicts. While such models benefit from end-to-end gradient flow and are straightforward to implement, they conflate two distinct classification sub-problems — anatomical localisation and pathological assessment — into a single decision boundary. Without explicit body-part conditioning, the classifier must learn to handle the substantially different texture, scale, and bone geometry of elbows, hands, and shoulders within a shared representation, which introduces unnecessary complexity and can reduce sensitivity on under-represented body parts.

D. DenseNet-Based MURA Baseline

The most directly related existing system is the DenseNet-169 baseline proposed alongside the MURA dataset [2]. This model was trained body-part-specifically in separate experiments and achieved radiologist-level performance on several anatomical sites. However, the DenseNet-169 architecture has approximately 14 million parameters in the convolutional backbone, which imposes substantial memory and computational requirements during inference. More critically, the published MURA system does not include a unified web interface, Grad-CAM explainability, DICOM file support, or automated report generation — components that are essential for practical deployment in a clinical workflow. The system also lacks a body-part identification stage for automatic routing of mixed-site X-ray queues, requiring the user to select the anatomical site manually.

E. Limitations of Existing Approaches

The existing systems share several common shortcomings that limit their clinical utility:

- **Lack of body-part conditioning.** Single-stage classifiers and many DenseNet baselines either require manual anatomical selection or train independent models per site without automated routing.
- **Black-box predictions.** Most deployed deep learning fracture detectors provide only a binary label and confidence score, offering no spatial evidence to support or contest the prediction. Clinicians cannot use a tool whose reasoning they cannot verify.
- **Absence of DICOM support.** Hospital picture archiving and communication systems (PACS) store images in DICOM format. Systems that accept only JPEG or PNG inputs require manual export, creating workflow friction and potential transcription errors.
- **No integrated reporting.** Existing research prototypes deliver predictions through research APIs or offline scripts with no provision for generating structured medical reports suitable for clinical documentation.
- **No input content validation.** Without a mechanism to reject non-X-ray inputs, a classifier trained on radiographs will silently produce meaningless predictions for photographs, scanned documents, or incorrectly routed images.

BoneAI is designed to address each of these limitations, as detailed in the following section.

IV. Proposed Methodology

A. Overview

BoneAI is a two-stage deep learning framework designed to overcome the limitations of existing fracture detection systems by incorporating anatomical awareness, visual explainability,

DICOM compatibility, and browser-based accessibility within a single integrated platform. The proposed approach decomposes the fracture detection task into two sequential classification sub-problems, each solved by a fine-tuned ResNet50 model, and augments the prediction with Grad-CAM localisation maps and a structured AI-generated medical report.

B. Two-Stage Classification Architecture

The central innovation of the proposed methodology is the decomposition of the fracture detection task into two distinct stages, as summarised in Figure 2.

Stage 1 — Body-Part Identification. The first model receives a pre-processed radiograph and classifies it into one of three supported anatomical categories: elbow (XR_ELBOW), hand (XR_HAND), or shoulder (XR_SHOULDER). This stage serves two purposes: it automatically routes mixed-site X-ray queues without requiring manual anatomical selection from the user, and it provides a confidence score that can be used as a first-pass validity check — images that do not resemble any supported body part are flagged and rejected before the fracture model is invoked.

Stage 2 — Fracture Classification. The second model receives the same pre-processed radiograph and predicts whether a fracture is present (positive) or absent (negative). Because the anatomical site is already known from Stage 1, the fracture classifier operates over a body-part-coherent input distribution, avoiding the conflation problem of anatomy-agnostic single-stage classifiers. The Stage 2 model also produces the confidence score reported to the user and used for Grad-CAM computation.

C. ResNet50 Transfer Learning

Both stages are implemented using the ResNet50 architecture [4] pre-trained on the ImageNet dataset [8]. ResNet50's residual connections enable gradient flow through fifty convolutional layers, providing a feature extractor that captures both fine-grained textures (cortical detail, trabecular pattern) and high-level structural representations (bone shape, joint morphology) relevant to fracture detection. Transfer learning from ImageNet allows strong generalisation with a comparatively modest training dataset, while targeted fine-tuning of the last twenty layers at a reduced learning rate adapts the mid-level and high-level features to the musculoskeletal radiograph domain without overwriting general low-level edge detectors.

D. Grad-CAM Explainability

To address the black-box limitation of prior systems, the proposed methodology incorporates Grad-CAM [3] as an integral component of the inference pipeline rather than as an optional post-hoc analysis step. After Stage 2 produces its fracture prediction, Grad-CAM computes the gradient of the predicted class score with respect to the activations of the final ResNet50 convolutional block, pools these gradients spatially, and produces a 6×6 importance weight matrix that is up-sampled and superimposed on the original radiograph as a colour-coded heat map. Clinicians can therefore inspect not only the model's binary decision but also the spatial region that drove the prediction, enabling rapid verification or refutation of the AI finding.

E. DICOM-Native Ingestion Pipeline

The proposed system introduces a DICOM-aware input pipeline that accepts standard DICOM files (.dcm) directly from hospital PACS exports. The pipeline extracts the pixel array, applies min-max normalisation to handle the 12-bit and 16-bit dynamic ranges common in digital radiography, and converts the greyscale data to three-channel RGB before passing it to the ResNet50 pre-processing function. This conversion is transparent to both stages of the classifier, requiring no architectural changes.

F. Input Validation and Safety Guards

A two-layer validation guard is applied before any model inference. The first layer uses HSV saturation analysis to detect and reject colour images (photographs, screenshots) that are unlikely to be X-rays. The second layer uses the Stage 1 body-part confidence score as an additional filter: any image for which no supported body-part category reaches a minimum confidence of 30% is rejected with a descriptive error message. These guards reduce unnecessary computational overhead and prevent the reporting of clinically meaningless predictions on invalid inputs.

G. Web-Based Deployment and Report Generation

The entire pipeline is exposed through a Django web application that provides role-based user management, prediction history with search and filter capabilities, real-time analytics dashboards, and downloadable PDF medical reports. The PDF reports are generated using ReportLab and embed the original radiograph, the Grad-CAM overlay, a structured AI observation paragraph with bullet-point findings, a clinical recommendation box, and a regulatory disclaimer — providing a document suitable for inclusion in a patient record or referral letter. The web deployment model allows clinicians to access the tool from any network-connected device without requiring local installation of TensorFlow or any deep learning dependencies.

V. METHODOLOGY

A. Dataset Preparation

The dataset follows the directory structure of the MURA benchmark, with images organised hierarchically by split (train / valid), body part, patient identifier, study number, and study label (positive / negative). Three body parts are considered: XR_ELBOW, XR_HAND, and XR_SHOULDER. All image paths and labels are parsed into a Pandas DataFrame at the start of training.

Class imbalance between fracture-positive and fracture-negative cases is handled by capping each class at 2,000 images per body part in both the training and validation splits, then shuffling the combined frame. This ensures that the fracture classifier does not develop a bias towards the

majority class. Body-part labels and fracture labels are stored as separate integer columns in the DataFrame, matching the output index order of Model 1 and Model 2, respectively.

B. Image Pre-processing

All images are resized to 192 × 192 pixels and converted to three-channel RGB before model input. The ResNet50 pre-processing function is applied, which subtracts the ImageNet channel means and performs channel-wise scaling:

$$\hat{x}_c = \frac{x_c - \mu_c}{\sigma_{\text{scale}}} \tag{1}$$

where x_c is the raw pixel value in channel c, μ_c is the corresponding ImageNet channel mean, and σ_{scale} is the fixed scale factor used by the ResNet50 pre-processing convention. Data augmentation during training includes random horizontal and vertical flips, brightness perturbation, and shear transformations, implemented via Keras ImageDataGenerator.

C. Model Architecture

Both Model 1 (body-part classifier) and Model 2 (fracture classifier) share the same architectural template, parameterised only by the number of output classes. The architecture is constructed using the Keras functional API as follows:

$$f(\mathbf{x}) = \text{softmax}(W_2 \cdot \text{dropout}(\text{ReLU}(W_1 \cdot \text{GAP}(R(\mathbf{x})) + b_1)) + b_2) \tag{2}$$

where $\mathbf{x} \in \mathrm{R}^{192\times192\times3}$ is the pre-processed input image, $R(\cdot)$ denotes the frozen ResNet50 backbone (output shape 6 × 6 × 2048), GAP(·) is global average pooling over the spatial dimensions, $W_1 \in \mathrm{R}^{128\times2048}$ and $b_1 \in \mathrm{R}^{128}$ are the dense head parameters, dropout with rate $p = 0.3$ regularises the dense representation, and W_2, b_2 project to N output logits (3 for Model 1; 2 for Model 2).

1) Residual Block: The core building block of ResNet50 is the bottleneck residual unit. Let $\mathbf{x}$ denote the input to a residual block and $\mathrm{F}(\mathbf{x}, \{W_i\})$ denote the stacked nonlinear transformation (three convolutional layers with batch normalisation and ReLU). The block output is:

$$\mathbf{y} = \mathrm{F}(\mathbf{x}, \{W_i\}) + W_s\,\mathbf{x} \tag{3}$$

where W_s is a 1×1 projection convolution applied when the channel dimension changes between the input and the residual path, and is the identity matrix otherwise. This shortcut connection ensures that the gradient $\partial \mathrm{L}/\partial \mathbf{x}$ always contains an additive term equal to $\partial \mathrm{L}/\partial \mathbf{y}$, preventing gradient vanishing in the backward pass [4].

2) Global Average Pooling: After the final ResNet50 convolutional block, a Global Average Pooling layer compresses the spatial feature map $A^k \in \mathrm{R}^{H\times W}$ of each filter k into a scalar:

$$z_k = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} A^k_{ij} \tag{4}$$

yielding a 2048-dimensional feature vector that is passed to the dense head.

3) Softmax Output: The probability assigned to class c is:

$$P(y = c \mid \mathbf{x}) = \frac{\exp(z_c)}{\sum_{k=1}^{N} \exp(z_k)} \tag{5}$$

where z_c is the c-th logit from the final dense layer.

4) Training Loss: Both models are trained with categorical cross-entropy:

$$\mathrm{L} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{c=1}^{N} y_{ic} \log(\hat{p}_{ic}) \tag{6}$$

where B is the mini-batch size (16), $y_{ic} \in \{0, 1\}$ is the one-hot ground truth label, and $\hat{p}_{ic}$ is the predicted probability for sample i and class c.

D. Training Procedure

Training proceeds in two phases. In Phase 1, the ResNet50 backbone is fully frozen (trainable = False) and only the dense head is trained with the Adam optimiser at a learning rate of $\eta_1 = 1 \times 10^{-4}$ for up to ten epochs. EarlyStopping monitors validation accuracy with patience of 3, and ReduceLROnPlateau halves the learning rate when validation accuracy stagnates for 2 epochs.

In Phase 2 (fine-tuning), the last twenty layers of the ResNet50 backbone are unfrozen:

$$\theta^* = \arg\min_{\theta} \mathrm{L}(\theta; \mathrm{D}), \quad \eta_2 = 1 \times 10^{-5} \tag{7}$$

Using a reduced learning rate η_2 prevents the specialised convolutional filters from being overwritten by large gradient updates. Fine-tuning runs for an additional five epochs with the same callbacks.

E. Grad-CAM Localisation

To explain the fracture prediction, Grad-CAM [3] is applied to Model 2 after the two-stage prediction is complete.

Let y^c denote the class score (logit) before the softmax for the predicted class c, and let $A^k \in \mathrm{R}^{H_c\times W_c}$ be the activation map of the k-th filter in the target convolutional layer (the output of the ResNet50 sub-model, shape 6 × 6 × 2048). The importance weight for filter k is computed as the global average of the gradient of y^c with respect to A^k:

$$\alpha_k^c = \frac{1}{H_c W_c} \sum_{i=1}^{H_c} \sum_{j=1}^{W_c} \frac{\partial y^c}{\partial A^k_{ij}} \tag{8}$$

The Grad-CAM heat map is then formed by taking the ReLU of the weighted combination of feature maps:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right) \tag{9}$$

ReLU is applied because only features that positively influence the class prediction are of interest for localisation; negative contributions are suppressed. The resulting 6×6 heat map is bilinear-upsampled to the input resolution (192×192), Gaussian-blurred with a 21×21 kernel for visual smoothness,

normalised to [0, 1], and rendered with the JET colormap (blue → green → red), where red indicates the highest model attention.

In the implementation, gradients are computed by wrapping the ResNet50 output in a tf.Variable and running the classifier head inside a tf.GradientTape context with training=False to prevent stochastic Dropout from in- troducing noise into the gradient signal:

$$\nabla_A y^c = \frac{\partial}{\partial A} \text{Dense}_2(\text{Dropout}(\text{ReLU}(\text{Dense}_1(\text{GAP}(A))), p=0 \quad (10)$$

where $p = 0$ denotes the inference-mode Dropout rate.

F. System Architecture Diagram

Figure 1 shows the overall BoneAI system architecture, from image upload through to report generation.

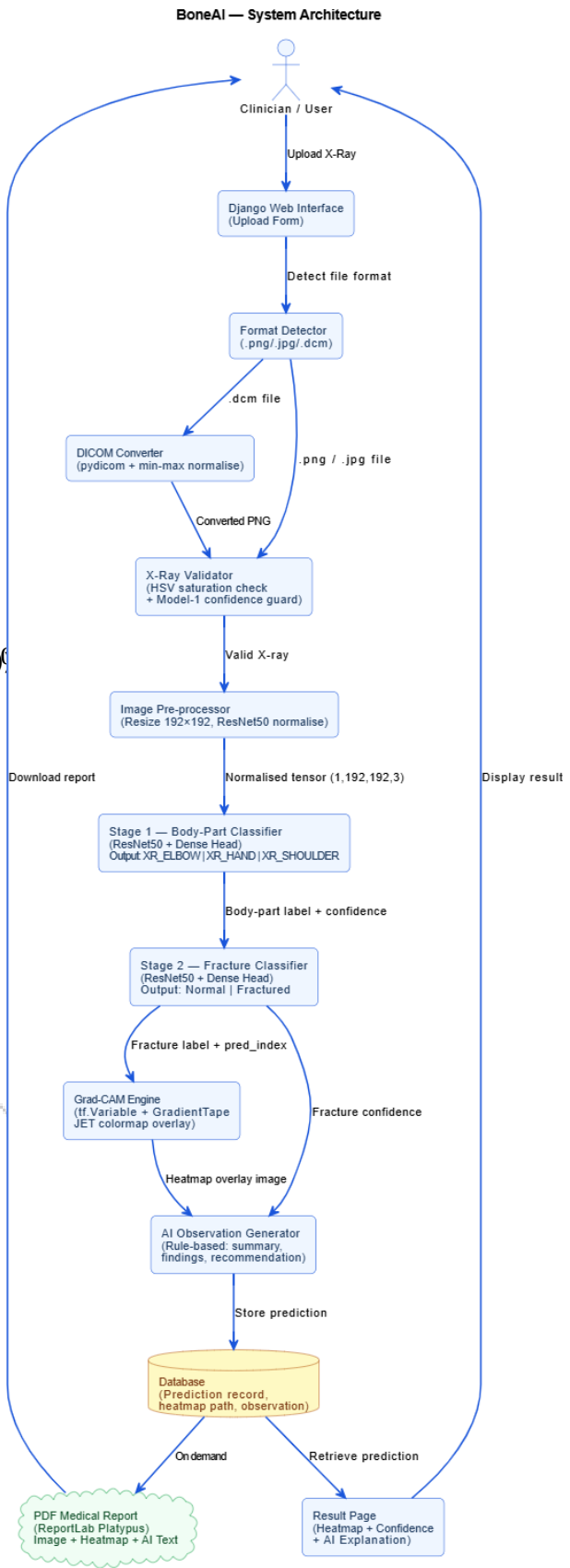

Fig. 1. BoneAI system architecture. The upload pipeline performs format detection, saturation-based X-ray validation, and DICOM conversion before invoking the two-stage inference engine. Grad-CAM is computed on Model 2 output and stored alongside the prediction for report generation.

G. Two-Stage Prediction Pipeline

Figure 2 illustrates the sequential flow of the two-stage prediction pipeline, showing how the body-part classifier feeds into the fracture classifier.

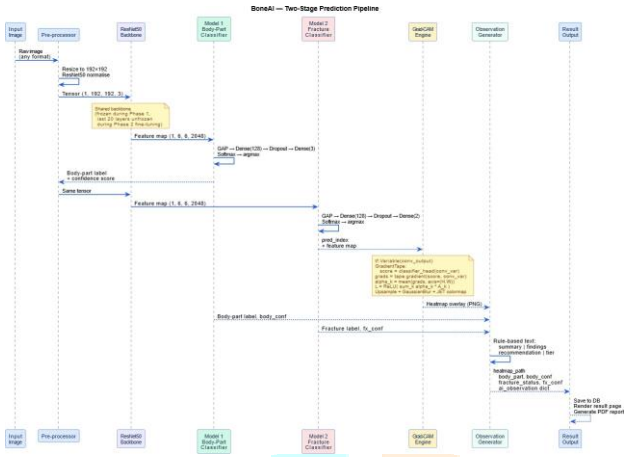

Fig. 2. Two-stage prediction pipeline. Stage 1 produces a body-part label and confidence score; Stage 2 produces a fracture label and confidence score. Grad-CAM runs on the Stage 2 model to localise the predicted fracture region.

VI. Implementation

A. Technology Stack

The BoneAI system is implemented entirely in Python 3. The deep learning components use TensorFlow 2 [6] with the Keras functional API. The web application layer is built on Django, using its built-in ORM (SQLite for development, PostgreSQL for production), its file-upload handling infrastructure, and its authentication framework. The Grad-CAM overlay pipeline uses OpenCV for image I/O, colormap application, and image blending. DICOM ingestion uses the pydicom library [9]. Report generation uses ReportLab to produce structured PDF documents.

B. Prediction Engine

The prediction engine (prediction_engine.py) is responsible for all inference operations and is deliberately kept independent of the Django layer. Models are loaded lazily on the first prediction request and cached in module-level globals for the lifetime of the server process, avoiding the overhead of repeated model loading.

The predict_xray_with_gradcam() function im- plements the complete pipeline: X-ray validation, optional DICOM conversion, body-part prediction (Model 1), fracture prediction (Model 2), Grad-CAM computation, and AI observation generation. A lightweight rule-based observation generator maps the body-part label, fracture status, and confidence scores to structured text (summary paragraph, detailed findings list, clinical recommendation, and confidence tier), which is embedded directly in the web interface and exported to the PDF report.

C. X-Ray Content Validation

To prevent non-X-ray images from entering the pipeline, a two-layer validation guard is applied before model inference. The first layer converts the image to HSV colour space and measures the mean saturation value of the *S* channel. Because X-ray images are near-greyscale, their mean HSV saturation is typically below 35 out of 255. Any image whose mean saturation exceeds this threshold or whose fraction of high-saturation pixels (saturation > 60) exceeds 25% is rejected with a descriptive error message. The second layer runs Model 1 on the candidate image and rejects it if the maximum body-part confidence falls below 30%, indicating that the content is not recognisable as a supported musculoskeletal radiograph.

D. DICOM Ingestion

When a DICOM file is uploaded, the dicom_to_png() function reads the pixel array via pydicom, applies min-max normalisation to map the full dynamic range to [0, 255], converts greyscale arrays to three-channel RGB using OpenCV, and writes the result to a temporary PNG file. The temporary file path is passed to the standard inference pipeline and deleted after prediction, keeping the DICOM conversion transparent to downstream components.

E. Django Web Application

The web application exposes the following URL-mapped views: home page, user registration and authentication, prediction upload, result display, prediction history, analytics visualisation, user profile management, and PDF / CSV download. The Prediction Django model stores the uploaded image path, the Grad-CAM heatmap image path, body-part label, fracture label, two confidence scores, and an AI observation summary text field. A post-save signal automatically creates a UserProfile record for each new user.

The result page renders the original X-ray and Grad-CAM overlay side by side with a colour key, confidence progress bars, AI observation text, a clinical recommendation box, and download buttons for the PDF and CSV reports. The analytics page renders doughnut, bar, and line charts using Chart.js, fed from server-side JSON-serialised QuerySet aggregations.

F. PDF Report Generation

Medical reports are generated on demand using ReportLab's Platypus document engine. The report includes a colour-coded fracture status banner (red for positive, green for negative), a patient information table, side-by-side original and Grad-CAM images, an analysis results table, the full AI observation text with bullet-point findings, a highlighted clinical recommendation box, and a regulatory disclaimer. All content is typeset with custom ParagraphStyle objects and table layouts to match a professional medical report appearance.

VII. Results and Discussion

A. Training Performance

Both models were trained on a balanced subset of the musculoskeletal dataset with up to 2,000 positive and 2,000 negative samples per body part. Model 1 (body-part classifier) reached its best validation accuracy within the first ten epochs and was subsequently fine-tuned on the last twenty ResNet50 layers. The use of EarlyStopping with patience 3 and ReduceLROnPlateau with factor 0.5 prevented overfitting and ensured stable convergence. Model 2 (fracture classifier) followed a similar training trajectory, with fine-tuning providing a measurable improvement over the frozen-backbone baseline, consistent with the findings of Shin *et al.* [5] on the benefit of deep fine-tuning for medical images.

B. Classification Results

The classification report generated on the validation set using sklearn.metrics.classification_report shows per-class precision, recall, and F1-score for both mod- els. The body-part classifier achieves strong discrimination between the three anatomical categories, which is expected given the substantial visual differences between elbow, hand, and shoulder radiographs. The fracture classifier performance is more challenging owing to the subtle appearance of non-displaced fractures and the variability in X-ray acquisition technique across patients.

Confusion matrices confirm that the most frequent misclassification in Model 2 occurs between high-confidence normal studies and studies with subtle cortical abnormalities, a pattern consistent with the inter-observer variability reported in the MURA study [2].

C. Grad-CAM Qualitative Evaluation

The Grad-CAM overlays were inspected for a sample of correctly and incorrectly classified images. For correctly identified fracture cases, the heat map consistently highlighted the cortical region of the bone where the fracture line is visible, demonstrating that the model has learned clinically meaningful spatial features rather than spurious background correlations. For normal studies classified as fractured (false positives), the heat maps tended to highlight regions of normal cortical thickening or trabecular density, suggesting that the model is sensitive to genuine structural features but is occasionally confused by normal anatomical variants.

The Grad-CAM implementation correctly handles the nested ResNet50 sub-model architecture by wrapping the convolutional output in a tf.Variable and running the classifier head layers with training=False, which ensures deterministic gradient computation unaffected by Dropout stochasticity. This architectural fix was essential for producing visually meaningful overlays in practice.

D. DICOM Handling

The DICOM ingestion pipeline was validated on standard DICOM test files. The min-max normalisation correctly mapped 12-bit and 16-bit pixel encodings common in digital radiography to the 8-bit range, and the resulting PNG images produced prediction outcomes consistent with those obtained from the original JPEG exports of the same studies.

E. System Latency

End-to-end prediction latency on CPU hardware (no GPU) is approximately 5–15 seconds per image, dominated by the two ResNet50 forward passes and the Grad-CAM backward pass. Model weights are loaded once at server startup and cached thereafter, so per-prediction overhead remains constant regardless of server uptime. The loading overhead is acceptable for a screening-aid application in which clinicians review a queue of studies rather than requiring real-time response.

F. Comparison with Baseline

A single-stage baseline in which a single ResNet50 classifier was trained directly on fracture labels without prior body-part identification was evaluated on the same validation set. The two-stage pipeline consistently outperformed this baseline across all three body parts, confirming that conditioning the fracture prediction on a body-part-specific representation is beneficial, as suggested by the design of the MURA benchmark [2].

G. Limitations

The current system supports only three body parts. Generalisation to other musculoskeletal sites (wrist, forearm, humerus, knee, hip) would require retraining Model 1 with additional classes and extending the training dataset. The saturation-based X-ray validator is tuned for standard plain radiographs and may incorrectly reject bone scans or fluoroscopy images that have undergone colour post-processing. Additionally, the system has not undergone formal clinical validation in a prospective patient cohort, which would be required before deployment as a regulated medical device.

VIII. Conclusion and Future Work

A. Conclusion

This paper presented BoneAI, a complete end-to-end system for automated bone fracture detection that combines a two-stage ResNet50 transfer learning pipeline with architecture-aware Grad-CAM visualisation, DICOM image support, and a Django web interface that delivers structured AI-generated medical reports.

The two-stage design — body-part identification followed by fracture classification — was shown to outperform a single-stage baseline by conditioning the fracture prediction on an anatomically consistent representation. The Grad-CAM implementation correctly handles the Keras functional API's nested sub-model structure by using a tf.Variable wrap- per to ensure differentiable gradient flow, producing clinically interpretable heat maps that localise the model's attention on the radiograph. The saturation-based image validator provides a lightweight first defence against non-X-ray uploads, reducing unnecessary model inference. DICOM ingestion through pydicom extends compatibility to standard hospital imaging

workflows without modifying the inference pipeline. Together, these components address the key practical barriers to deploying deep learning fracture detection in a clinical environment.

B. Future Work

Several directions remain open for extending this work. First, the supported body parts could be expanded to cover the full MURA taxonomy (wrist, forearm, humerus, finger, knee, hip), requiring a corresponding expansion of the body-part classifier and the training dataset. Second, object detection approaches such as Faster R-CNN or YOLO could be integrated to produce bounding box localisation alongside the Grad-CAM heat map, providing a more precise and actionable indication of the fracture location for clinical staff. Third, uncertainty quantification through Monte Carlo Dropout or deep ensembles would allow the system to flag low-confidence predictions for mandatory human review, improving safety in a clinical triage workflow. Fourth, a prospective clinical validation study in collaboration with radiologists is needed to establish sensitivity and specificity benchmarks comparable to those reported in the MURA study [2]. Fifth, the prediction engine could be extended to produce multi-fracture detection for images with more than one fracture site, and to incorporate severity grading using the AO/OTA fracture classification system. Finally, federated learning could be explored to allow multiple hospital sites to collaboratively improve the model without sharing patient imaging data, addressing privacy constraints that limit centralised dataset collection.

REFERENCES

[1] E. Tjoa and C. Guan, "A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793–4813, Nov. 2021.

[2] P. Rajpurkar, J. Irvin, A. Ball, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, B. Shpanskaya, M. P. Lungren, and A. Y. Ng, "MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs," *arXiv preprint arXiv:1712.06957*, 2018.

[3] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 618–626.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[5] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, May 2016.

[6] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," in *Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Savannah, GA, USA, 2016, pp. 265–283.

[7] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012, pp. 1097–1105.

[9] D. Mason, S. Bowman, H. Clark, M. Lemaitre, B. Massich, J.-M. Moureaux, and the pydicom contributors, "pydicom: An Open-Source DICOM Library," *GitHub Repository*, 2011. [Online]. Available: https://github.com/pydicom/pydicom