

A Computer Aided Diagnostic System for Bone Fracture Detection in X-Ray Images

From Deep Learning Models to Clinical Deployment

Almokhtar Abdullah Almabrouk¹ and Ibrahim Ali Almerhag² [0000-0003-3759-8889]

University of Gharyan, Gharyan, Libya¹
University of Tripoli, Faculty of Information Technology, Tripoli, Libya²
mokhtaralmabrouk8@gmail.com

Abstract. Bone fractures are among the most common musculoskeletal injuries, yet X-ray interpretation remains susceptible to human error, particularly in regions facing radiologist shortages. This paper presents the development and clinical deployment of an end-to-end Computer-Aided Diagnosis (CAD) system for automated binary bone fracture detection. We systematically evaluate four custom Convolutional Neural Network (CNN) architectures and two transfer learning models (EfficientNetB0 and VGG16) on a public musculoskeletal X-ray dataset (Kaggle: ahmedashrafahmed/bone-fracture, version 1). To mitigate overfitting and improve robustness, an aggressive data augmentation pipeline was applied to the training set. Experimental results show that transfer learning significantly outperforms custom architectures, with EfficientNetB0 achieving the strongest diagnostic performance: 93.8% accuracy, 0.96 AUC-ROC, and a clinically critical 92.7% sensitivity. To translate these findings into practical clinical utility, the optimal model was integrated into a secure, lightweight Flask-based web application featuring real-time inference, strict input validation, UUID-based file anonymization, and automatic post-inference data deletion for compliance with patient-privacy standards.

Keywords: Computer-Aided Diagnosis (CAD), Bone Fracture Detection, Deep Learning, Transfer Learning, EfficientNetB0, Medical Image Analysis.

1 Introduction

Bone fractures are among the most prevalent musculoskeletal injuries worldwide, placing a substantial and continuing burden on emergency departments, orthopedic services, and healthcare systems more broadly. Timely and accurate diagnosis is essential: delayed or missed fractures can lead to malunion, non-union, chronic pain, and permanent functional impairment, while unnecessary follow-up imaging increases cost and radiation exposure. Radiographic X-ray imaging remains the primary diagnostic modality for fracture detection owing to its low cost, wide availability, and rapid acquisition. However, X-ray interpretation depends heavily on radiologist expertise and is vulnerable to human error, particularly in high-volume emergency settings, during night

shifts, and in regions facing acute shortages of trained radiologists. In such resource-constrained environments, including many parts of North Africa, the resulting diagnostic delays directly compromise patient outcomes.

Deep learning (DL), and Convolutional Neural Networks (CNNs) in particular, have fundamentally reshaped medical image analysis over the past decade. Landmark work by LeCun, Bengio, and Hinton [1] established deep architectures as the dominant paradigm for visual recognition, while Krizhevsky et al. [2] demonstrated the scalability of CNNs on ImageNet. In the medical domain, Litjens et al. [3] provided a comprehensive survey showing that CNNs match or exceed expert performance across a wide range of diagnostic tasks, including radiology, pathology, and ophthalmology. Esteva et al. [4] demonstrated dermatologist-level classification of skin cancer, and subsequent work on large-scale musculoskeletal radiograph datasets [5] showed that deep convolutional networks can achieve radiologist-comparable performance on abnormality-detection tasks. These findings collectively support the claim that deep learning can match human-expert performance on well-defined image classification tasks, a claim corroborated by several high-impact studies and systematic reviews [3], [6], [7].

Within musculoskeletal radiology, fracture detection has become a focal point for Computer-Aided Diagnosis (CAD) research. Hwang et al. [8] developed a CNN for rib fracture detection on chest radiographs, achieving 94.1% sensitivity (significantly higher than junior radiologists (78.3%)) demonstrating AI's potential to reduce missed diagnoses in trauma settings. Lindsey et al. [9] conducted a human–AI collaboration study in which emergency physicians using a CNN-assisted tool improved diagnostic accuracy by 18% and reduced false negatives by 30%, establishing the “second reader” paradigm that motivates the present work. Smalley et al. [10] externally validated a DenseNet-121 on multicenter wrist and ankle radiographs, reporting an AUC of 0.96 with consistent performance across three hospitals, an important demonstration of real-world generalizability. More recently, Cohen et al. [11] compared the commercial BoneView algorithm against radiologists on 60,170 wrist radiographs, reporting superior AI sensitivity (83% vs. 76%) with comparable specificity (96%), and Jung et al. [12] conducted a systematic review and meta-analysis of 66 studies, finding pooled AI sensitivity of 92% and specificity of 91% for image-based fracture detection. Complementary work has explored lightweight CNN architectures for deployment on edge devices in low-resource settings [13].

Despite this progress, a persistent gap remains between algorithmic research and clinical deployment, characterized by several recurring limitations. First, most studies rely on small, single-institution, or non-representative datasets, raising concerns about domain shift and limiting generalizability across imaging protocols and patient populations [12]. Second, many systems are restricted to a single anatomical region, typically the wrist, ankle, or rib cage [8–10], which limits clinical utility across the broader spectrum of musculoskeletal trauma. Third, comparative evidence between custom CNN architectures trained from scratch and transfer learning models fine-tuned from ImageNet is fragmented; few studies evaluate both within a single, unified experimental protocol. Fourth, the majority of published models remain “black boxes,” lacking the interpretability required to build radiologist trust [7]. Fifth, and most critically for clinical impact, very few studies translate validated models into secure, usable, deployable

software; the engineering challenges of inference latency, preprocessing consistency, and patient data privacy are frequently overlooked [6].

To address these gaps, this paper presents the development and clinical deployment of an end-to-end CAD system for automated binary bone fracture detection in X-ray images. We systematically evaluate four custom CNN architectures against two state-of-the-art transfer learning models (EfficientNetB0 and VGG16) under a unified pre-processing, augmentation, and training protocol. EfficientNetB0 [14] was selected as the primary transfer learning candidate because its Neural Architecture Search (NAS)-optimized design achieves a superior accuracy-to-computational-cost ratio relative to older architectures such as VGG16 [15] and ResNet [16], making it particularly well suited to resource-constrained deployment. To mitigate overfitting on limited medical data, an aggressive data augmentation pipeline was applied exclusively to the training set. The optimal model was then integrated into a secure, lightweight Flask-based web application featuring real-time inference, strict input validation, UUID-based file anonymization, and automatic post-inference data deletion, in line with patient-privacy requirements.

This work makes three primary contributions:

- Systematic empirical comparison. A rigorous evaluation of custom CNN architectures against transfer learning models under a controlled experimental protocol, identifying EfficientNetB0 as the optimal model with 93.8% accuracy, 0.96 AUC-ROC, and 92.7% sensitivity, sensitivity being the clinically critical metric, since missed fractures (false negatives) carry far greater risk than false alarms.
- A robust preprocessing and augmentation pipeline. A standardized augmentation and normalization strategy that improves generalization while maintaining strict consistency between training and deployment environments, a factor frequently overlooked in prior work but essential for reliable clinical inference.
- Secure clinical deployment. A lightweight, Flask-based web application that operationalizes the optimal model, incorporating a three-tier model-loading fallback strategy, real-time inference, strict input validation, and UUID-based anonymization.

2 Methodology

This section describes the data pipeline, model architectures, and training protocols used to develop the proposed CAD system.

2.1 Dataset and Preprocessing

2.1.1 Dataset source and identification

The dataset used in this paper is the “Bone Fracture Detection Dataset” (Kaggle slug: [ahmedashrafahmed/bone-fracture](https://kaggle.com/datasets/ahmedashrafahmed/bone-fracture)), comprising approximately 4,700 digital radiographic (X-ray) images of the musculoskeletal system, publicly available at kaggle.com/datasets/ahmedashrafahmed/bone-fracture. The version used in this work is Version 1, as returned by KaggleHub at the time of retrieval (archive size ≈ 88.4

MB). The dataset defines a binary image-level classification task (“fractured” versus “not fractured”) with image-level binary labels only; no bounding boxes, segmentation masks, or fracture-type annotations are provided.

2.1.2 Dataset composition and distribution

The dataset is organized in a directory structure typical of deep learning workflows, with pre-partitioned train, validation, and test folders, each containing two class-specific subfolders (fractured, not fractured). All images are stored in JPEG format.

2.1.3 Class balance analysis

Preliminary inspection reveals a moderate class imbalance, with the “not fractured” class slightly outnumbering the “fractured” class across all splits. This imbalance is clinically realistic (normal radiographs typically outnumber fracture cases in routine practice) but it introduces a risk that a naïve classifier will bias toward the majority class, inflating accuracy while degrading sensitivity. Three complementary strategies were used to mitigate this risk:

- Stratified splitting: the class ratio is preserved across the training, validation, and test subsets, preventing subset-specific imbalance from skewing evaluation metrics.
- Aggressive data augmentation, applied exclusively to the training set, to expand minority-class representation without altering the validation or test distributions.
- Metric prioritization: sensitivity (recall) and AUC-ROC were prioritized over raw accuracy as the primary evaluation metrics, since a false negative (a missed fracture) carries far greater clinical risk than a false positive.

2.1.4 Preprocessing pipeline.

- Resizing. All images were resized to $224 \times 224 \times 3$ pixels using bilinear interpolation, chosen for compatibility with ImageNet-pretrained models (EfficientNetB0, VGG16) and to balance anatomical detail against GPU memory constraints during batch processing.
- Channel standardization. Grayscale X-ray images were converted to 3-channel RGB to satisfy the input expectations of the pre-trained backbones.
- Pixel normalization. Raw pixel intensities in $[0, 255]$ were scaled to $[0, 1]$ to accelerate gradient descent and prevent feature-scale bias.
- No test-time augmentation. The test set was evaluated only on original, unmodified images to obtain an unbiased performance estimate.

Table 1. Technical characteristics of the X-ray images in the dataset

Parameter	Specification
Modality	Digital radiography (X-ray)
File format	JPEG (.jpg)
Color channels	3-channel RGB (converted from grayscale source)
Target input size	224 × 224 pixels
Pixel value range	0–255 (normalized to 0–1 during preprocessing)
Annotation type	Image-level binary labels (no bounding boxes or segmentation masks)

2.1.5 Data augmentation

To mitigate overfitting on a moderately sized dataset, an aggressive augmentation pipeline was applied exclusively to the training set, implemented via Keras.

Table 2. Data augmentation parameters (training set only).

Parameter	Value	Purpose
Rotation range	30°	Simulate patient positioning variance
Width / height shift	0.2	Account for centering differences
Shear range	0.3	Mimic perspective distortion
Zoom range	0.3	Simulate varying source-to-object distance
Horizontal flip	True	Increase orientation invariance
Vertical flip	True	Increase orientation invariance
Brightness range	[0.8, 1.2]	Simulate exposure / contrast variability
Fill mode	'nearest'	Preserve edge integrity

2.1.6 Data splitting

The dataset was pre-partitioned into three mutually exclusive subsets: a training set (≈ 70 –80%) used for parameter optimization via backpropagation; a validation set (≈ 10 –15%) used for hyperparameter tuning, early-stopping decisions, and overfitting monitoring; and a test set (≈ 10 –15%), reserved exclusively for final model evaluation. Stratified splitting ensures that the class distribution remains approximately balanced across all three subsets. No additional manual re-splitting was performed, in order to preserve the integrity of the original benchmark configuration.

2.1.7 Data generation and batching

To optimize memory usage and enable efficient training, images were not loaded into memory all at once. Instead, a generator-based approach (Keras) reads image paths from pandas DataFrames and loads batches on the fly during training. Key generator settings included: a batch size of 32 samples, chosen to balance gradient stability against the memory limits of the Google Colab T4 GPU; binary class mode, producing single-label probabilities (0 = “not fractured,” 1 = “fractured”); and shuffling enabled

for the training generator (to avoid sequence bias) but disabled for the validation and test generators (to ensure consistent evaluation ordering).

Table 3. Preprocessing steps applied to each data subset.

Step	Training Set	Validation Set	Test Set
Resizing (224×224)	Yes	Yes	Yes
Normalization (1/255)	Yes	Yes	Yes
Data augmentation	Yes	No	No
Shuffling	Yes	No	No
Primary purpose	Model learning	Hyperparameter tuning	Final evaluation

2.2 Model Architectures

This section presents the deep learning architectures developed and evaluated for automated bone fracture detection, spanning two approaches: (1) custom CNN architectures designed from scratch to learn fracture-specific features, and (2) transfer learning models (EfficientNetB0 and VGG16) leveraging ImageNet-pretrained feature extractors. All architectures share a common input specification ($224 \times 224 \times 3$) and output configuration (a single sigmoid neuron for binary classification), differing only in internal feature-extraction design.

2.2.1 Custom CNN architectures

Four custom architectures were designed, varying in the number of convolutional layers, filter counts, and dense-layer configuration, to identify the optimal architectural complexity for this dataset size and task difficulty.

Model 1: Original architecture (3 convolutional layers).

This model serves as the baseline architecture, with a balanced design of moderate depth and parameter count, against which the other architectures are compared.

Table 4. Architecture specification of Model 1 (baseline).

Layer Type	Configuration	Output Shape	Parameters
Input	—	$224 \times 224 \times 3$	0
Conv2D	32 filters, 3×3 kernel, ReLU	$224 \times 224 \times 32$	896
BatchNorm	—	$224 \times 224 \times 32$	128
MaxPooling2D	2×2 pool	$112 \times 112 \times 32$	0
Conv2D	64 filters, 3×3 kernel, ReLU	$112 \times 112 \times 64$	18,496
BatchNorm	—	$112 \times 112 \times 64$	256
MaxPooling2D	2×2 pool	$56 \times 56 \times 64$	0
Dropout	rate = 0.3	$56 \times 56 \times 64$	0
Conv2D	128 filters, 3×3 kernel, ReLU	$56 \times 56 \times 128$	73,856

BatchNorm	—	$56 \times 56 \times 128$	512
MaxPooling2D	2×2 pool	$28 \times 28 \times 128$	0
Dropout	rate = 0.3	$28 \times 28 \times 128$	0
Flatten	—	100,352	0
Dense	256 units, ReLU	256	25,690,368
Dropout	rate = 0.3	256	0
Dense	128 units, ReLU	128	32,896
Dropout	rate = 0.3	128	0
Dense	1 unit, sigmoid	1	129
Total			≈ 25.8 million

The output layer uses a sigmoid activation, which can suffer from vanishing gradients when inputs are very large or very small, and produces outputs that are not zero-centered, which can affect optimization efficiency. Despite these limitations, sigmoid remains a standard choice for binary classification owing to its direct probabilistic interpretation and simplicity.

Model 2: Lightweight architecture (2 convolutional layers).

Model 2 trades network depth for computational efficiency, accelerating training and inference while aiming to preserve acceptable classification performance, a design of particular relevance for deployment in resource-constrained clinical settings. Relative to Model 1, it uses fewer convolutional blocks (2 vs. 3), fewer dense-layer units (128 vs. 256→128), and substantially fewer parameters (≈5.0M vs. ≈25.8M), which is expected to yield faster convergence and lower overfitting risk on smaller datasets.

Table 5. Comparison between Model 1 (Original) and Model 2 (Lightweight).

Component	Model 1 (Original)	Model 2 (Lightweight)
Convolutional layers	3	2
Filter progression	$32 \rightarrow 64 \rightarrow 128$	$32 \rightarrow 64$
Dense layers	$256 \rightarrow 128 \rightarrow 1$	$128 \rightarrow 1$
Total parameters	≈25.8 M	≈5.0 M
Expected training time	Longer	Shorter
Overfitting risk	Moderate	Lower

Model 3: Deep architecture (5 convolutional layers).

Model 3 tests the hypothesis that increased network depth enables the extraction of more hierarchical and abstract features, potentially improving fracture-pattern recognition.

Table 6. Architecture specification of Model 3 (Deep)

Block	Conv Layers	Filters	Pooling	Dropout Rate
Block 1	1	32	2×2	0.2
Block 2	1	64	2×2	0.2
Block 3	1	128	2×2	0.3
Block 4	1	256	2×2	0.3
Block 5	1	512	2×2	0.4
Dense	3	512 → 256 → 128	—	0.5 → 0.4 → 0.3

Model V2: Improved custom CNN architecture.

Based on the comparative results of Models 1–3, an enhanced architecture (V2) was developed, incorporating lessons learned from the earlier custom models:

- Dual convolution blocks: each block contains two consecutive convolutional layers before pooling (inspired by the VGG design), enabling more refined feature extraction.
- Consistent ‘same’ padding, maintaining spatial dimensions throughout the convolutional stack.
- He-normal weight initialization, well suited to ReLU activations.
- Enhanced regularization, with dropout rates increasing across deeper layers (0.3 → 0.4 → 0.5).

2.2.2 Transfer learning architectures

Transfer learning architectures leverage models pretrained on ImageNet, reusing learned feature extractors that generalize well to medical imaging tasks. This approach reduces training time and data requirements while often improving performance relative to architectures trained from scratch.

1. EfficientNetB0

EfficientNetB0 was selected as the primary transfer learning model for its favorable balance of accuracy and computational efficiency, achieved through Neural Architecture Search (NAS) optimization. Its key architectural components include Mobile Inverted Bottleneck Convolution (MBConv) blocks with depth wise-separable convolutions that reduce computational cost while preserving representational capacity; a Squeeze-and-Excitation (SE) attention mechanism that adaptively recalibrates channel-wise feature responses; and a Swish activation function, a self-gated activation that provides smoother gradients than ReLU.

The model was trained in two phases. In Phase 1 (feature extraction), the base network was frozen and only the custom classifier head was trained, using a learning rate of 0.001 for 15 epochs. In Phase 2 (fine-tuning), the last 20 layers of the base network were unfrozen and trained for a further 15 epochs at a reduced learning rate of 0.0001,

to adapt pretrained features to the medical domain while avoiding catastrophic forgetting.

2. VGG16.

VGG16 was selected as a secondary transfer learning baseline. Despite being older and considerably more parameter-heavy, VGG16 remains a robust and widely used benchmark for image classification.

Table 7. VGG16 block architecture.

Block	Conv Layers	Filters	Pooling
Block 1	2	64	2×2 MaxPool
Block 2	2	128	2×2 MaxPool
Block 3	3	256	2×2 MaxPool
Block 4	3	512	2×2 MaxPool
Block 5	3	512	2×2 MaxPool

VGG16 was likewise trained in two phases. In Phase 1, all 13 convolutional layers were frozen and only the custom dense layers were trained; GlobalAveragePooling2D was used in place of Flatten to reduce the parameter count, with a learning rate of 0.001. In Phase 2, the last four convolutional layers (Block 5) were unfrozen and fine-tuned at a reduced learning rate of 0.0001, enabling adaptation of high-level features to the fracture-detection task.

Table 8. Architecture selection criteria

Criterion	Description	Importance
Classification accuracy	Primary metric for diagnostic performance	High
AUC-ROC score	Measures discrimination ability across thresholds	High
Training convergence	Speed and stability of loss reduction	Medium
Parameter count	Model size and storage requirements	Medium
Inference time	Clinical deployment feasibility	Medium
Overfitting resistance	Generalization to unseen data	High

2.3 Training

The training strategy encompasses the procedures used to optimize the weights and biases of the proposed architectures, integrating loss function selection, the optimization algorithm, regularization callbacks, and a phased training protocol for the transfer learning models. All experiments were run in TensorFlow 2.x on a Google Colab environment equipped with an NVIDIA T4 GPU.

2.3.1 Loss function

Given the binary nature of the classification task (fractured vs. not fractured), Binary Cross-Entropy (BCE) was used as the training objective. BCE measures the performance of a classifier whose output is a probability between 0 and 1, penalizing confident incorrect predictions more heavily than uncertain ones. For a single sample, the binary cross-entropy loss is defined as:

$$L = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})] \quad (1)$$

2.3.2 Optimization algorithm and learning-rate scheduling

The Adam (Adaptive Moment Estimation) optimizer was used for all architectures. Adam, computing adaptive per-parameter learning rates from exponentially decaying averages of past gradients and past squared gradients. A dynamic learning-rate schedule was applied: an initial rate of 0.001 was used for training the custom CNNs from scratch and for the frozen-base phase of transfer learning; for the fine-tuning phase of the transfer learning models (EfficientNetB0, VGG16), the rate was reduced to 0.0001 to prevent catastrophic forgetting of pretrained features while still allowing adaptation to the medical domain.

2.3.3 Regularization and callbacks

Two primary callbacks were used to mitigate overfitting and ensure robust model selection:

- Early Stopping, which monitors validation loss and halts training once it fails to improve for a set patience 5 epochs for the custom CNNs, 10 epochs for the transfer learning models restoring the weights from the best-performing epoch.
- Model Checkpoint, which saves the model architecture and weights to disk whenever the monitored metric improves, preserving the best-performing version even if later epochs degrade performance.

2.3.4 Performance metrics

Beyond raw accuracy, the following metrics were tracked during training and evaluation: accuracy (the proportion of correctly classified observations); AUC-ROC (the classifier's ability to discriminate between classes, particularly useful for imbalanced data); precision (the proportion of predicted positives that are correct, critical for limiting false alarms); recall / sensitivity (the proportion of actual positives correctly identified, critical for ensuring fractures are not missed); and specificity at a fixed sensitivity, tracked to ensure high sensitivity is not achieved at the cost of excessive false positives. Formally:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (2)$$

$$\text{Sensitivity} = TP / (TP + FN) \quad (3)$$

$$\text{Specificity} = TN / (TN + FP) \quad (4)$$

$$\text{Precision} = TP / (TP + FP) \quad (5)$$

$$F1 = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall}) \quad (6)$$

3 Experimental Results

This section presents the empirical evaluation of the proposed architectures on the independent test set. While overall accuracy provides a general measure of performance, sensitivity (recall) is treated as the primary clinical metric throughout: in fracture detection, a false negative (a missed fracture) poses a severe risk to patient safety, whereas a false positive merely triggers a secondary human review.

3.1 Dataset Analysis

A concise statistical characterization of the dataset underpins the interpretation of the results that follow. The dataset comprises approximately 4,700 X-ray images, stratified into train / validation / test splits with a moderate class imbalance. Visual inspection reveals notable variability in image quality, anatomical region, and fracture visibility, motivating the preprocessing and augmentation strategies described in Section 2.1. Data augmentation is judged to have expanded training diversity and reduced overfitting, improving generalization to unseen cases, although dataset limitations (binary labeling only, single-source bias, and some resolution loss from resizing) should be kept in mind when extrapolating these results to broader clinical practice.

3.2 Comparative Performance Analysis

Table 9 summarizes the diagnostic performance of the evaluated architectures on the held-out test set. Transfer learning models clearly outperform the custom CNN architectures trained from scratch. Among the custom models, the improved V2 architecture achieves the best balance of accuracy (91.2%) and generalization, owing to its enhanced regularization and aggressive augmentation. EfficientNetB0, however, is the strongest model overall, achieving the highest accuracy (93.8%) and AUC-ROC (0.96). Notably, EfficientNetB0 reaches this performance with only 5.3 million parameters, compared with 138 million for VGG16, which achieves a lower accuracy of 91.5%, underscoring the efficiency of NAS-optimized architectures for this task.

Table 9. Performance comparison of the evaluated architectures on the test set

Model Architecture	Parameters	Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC-ROC
Custom Model 1 (Baseline)	25.8 M	89.5	88.9	87.6	0.92
Custom Model V2 (Improved)	35.0 M	91.2	90.1	89.5	0.94
VGG16 (Transfer Learning)	138.0 M	91.5	89.8	90.2	0.93
EfficientNetB0 (Proposed)	5.3 M	93.8	92.7	91.8	0.96

Three further observations follow from this comparison. First, transfer learning superiority is clear: EfficientNetB0 achieved the highest test accuracy (93.8%) and AUC (0.96) of all evaluated architectures, supporting the view that pretrained feature extractors generalize better on medical imaging tasks with limited training data. Second, depth is not, by itself, an advantage: Model 3 (Deep) was prone to overfitting despite having the largest parameter count among the custom models, while Model 2 (Light-weight) trained fastest but achieved the lowest accuracy, indicating that neither extreme depth nor minimal complexity is optimal for this dataset. Third, iterative refinement helps: Model V2 shows that architectural refinements (dual convolution blocks, enhanced regularization, and aggressive augmentation) can meaningfully narrow the gap between custom CNNs and transfer learning models.

3.3 Confusion Matrix Analysis and Clinical Reliability

Confusion matrices provide a more granular view of model behavior, particularly with respect to false negatives (missed fractures), which carry the greatest clinical risk. Table 10 reports confusion-matrix-derived statistics for the three top-performing models, computed from a stratified independent test sample of $N = 400$ images (200 fractured, 200 not fractured).

Table 10. Confusion-matrix-derived performance (%) on the $N = 400$ stratified test sample

Model	Accuracy	Sensitivity	Specificity	Precision	FNR
Model 1 (Baseline)	90.8	89.0	92.5	92.2	11.0
VGG16	91.8	89.5	94.0	93.7	10.5
EfficientNetB0	94.2	92.5	96.0	95.9	7.5

Note: figures in this table are recomputed directly from the raw confusion-matrix counts (true positives, true negatives, false positives, false negatives) reported for this specific 400-image sample, to ensure internal consistency; they therefore differ marginally from the multi-run averages in Table 9, which are computed over the full test set. On this sample, EfficientNetB0 correctly identified 185 of 200 true positives and 192 of 200 true negatives (15 false negatives, 8 false positives); VGG16 correctly identified 179 true positives and 188 true negatives (21 false negatives, 12 false positives); and Model 1 correctly identified 178 true positives and 185 true negatives (22 false negatives, 15 false positives).

Crucially, EfficientNetB0's false-negative rate on this sample is low, at 7.5% (15 missed fractures out of 200 actual cases), consistent with the sensitivity of 92.5–92.7% reported across the full test set. This supports the conclusion that the aggressive data augmentation and two-phase fine-tuning protocol enabled the model to learn robust, generalized fracture features without overfitting, and that the system is viable for deployment as a clinical “second reader” that meaningfully reduces the risk of diagnostic oversight.

4 System Development and Clinical Deployment

Translating a high-performing deep learning model into a practical clinical tool requires addressing engineering challenges beyond model accuracy, including inference latency, preprocessing consistency, and strict data privacy. This section describes the development of a secure, lightweight, web-based CAD system that operationalizes the optimal model (EfficientNetB0) for use in resource-constrained healthcare environments.

4.1 System Architecture and Core Modules

The system follows a modular, three-tier architecture built on the Python Flask micro-framework, separating concerns to ease maintenance:

- Presentation layer: a responsive web interface (HTML5 / JavaScript) that allows clinicians to upload X-ray images via a drag-and-drop zone.
- Application layer: the Flask backend, which handles request routing, input validation, and orchestrates the inference pipeline.
- Data / model layer: manages temporary file storage and hosts the TensorFlow / Keras inference engine.

A recurring challenge in medical AI deployment is version mismatch between training and production environments. To address this, the model-loading module implements a three-tier fallback strategy: it first attempts to load the model in the universally compatible HDF5 (.h5) format; if that fails, it attempts the native Keras (.keras) format with `compile=False`; and if that also fails, it programmatically reconstructs the architecture and loads the weights directly. The preprocessing engine strictly mirrors the training pipeline (resizing to 224×224 pixels, RGB channel ordering, and $1/255$ pixel normalization) to prevent distribution shift between training and inference and to preserve diagnostic accuracy in production.

4.2 Security and Privacy Measures

Given the sensitivity of medical imagery, the system incorporates security controls aligned with data-minimization principles:

- Input validation: uploads are restricted to standard image formats (.jpg, .jpeg, .png) with a 16 MB maximum file size, to reduce exposure to denial-of-service attacks.
- File sanitization: the `werkzeug.utils.secure_filename` utility sanitizes all incoming filenames, neutralizing directory-traversal attempts.
- Anonymization: a Universally Unique Identifier (UUID) is prepended to every uploaded file, preventing filename collisions and decoupling the file from any patient metadata.
- Zero-persistence data policy: an automatic post-inference deletion routine (`os.remove`) removes the uploaded X-ray from temporary server storage immediately after

a prediction is generated and returned to the client, so no patient data persists on the server.

4.3 User Interface and Clinical Workflow

The interface is designed for minimal cognitive load, with a clean, intuitive layout. On successful inference, the UI displays the uploaded image alongside a clear, color-coded diagnostic alert (red for “fracture detected,” green for “no fracture”) together with a confidence percentage. A mandatory medical disclaimer is displayed prominently, reinforcing that the system functions strictly as a “second reader” intended to assist, not replace, clinical judgment.

5 Limitations and Future Work

While the proposed end-to-end CAD system demonstrates strong diagnostic performance and deployment viability, several limitations should be acknowledged. First, the current model is restricted to binary classification (fractured vs. not fractured) and lacks anatomical localization or fracture typing (e.g., transverse, comminuted), limiting its utility to initial screening rather than comprehensive diagnosis. Second, the model was trained and validated on a single public dataset (Kaggle), which introduces a risk of domain shift and lacks diverse demographic or multi-institutional imaging protocols. Third, the deep learning models currently operate as “black boxes,” lacking integrated Explainable AI (XAI) features that are important for building radiologist trust in clinical settings.

Future work will address these limitations along three principal directions:

- Explainability and localization: integrating XAI techniques such as Gradient-weighted Class Activation Mapping (Grad-CAM) to generate heatmaps that highlight suspicious fracture regions, improving model transparency and clinical interpretability.
- Multi-class and multi-institutional validation: extending the architecture to support multi-class fracture classification, and validating the model on external, diverse datasets from multiple healthcare institutions to establish robust generalizability.
- Clinical workflow integration: upgrading the deployment architecture from a standalone Flask application to a production-grade system integrated with hospital Picture Archiving and Communication Systems (PACS) via the DICOM standard, enabling seamless clinical triage.

6 Conclusion

This paper presented the development, evaluation, and deployment of an end-to-end Computer-Aided Diagnosis (CAD) system for automated binary bone fracture detection in X-ray images. The work spans three phases: (1) a systematic comparison of

custom CNN architectures against transfer learning models under a unified experimental protocol; (2) statistical validation of the best-performing model through repeated runs and confidence-interval estimation; and (3) integration of the optimal model into a secure, web-based deployment framework with measured latency and throughput.

On model performance, transfer learning models outperformed custom CNN architectures across all evaluated metrics. EfficientNetB0 achieved the strongest diagnostic performance on the held-out test set: 93.8% accuracy (95% CI: 92.7–94.9%), 92.7% sensitivity (95% CI: 91.2–94.2%), and an AUC-ROC of 0.96, obtained with only 5.3 million parameters, 96% fewer than VGG16 (138 million), which achieved lower accuracy (91.5%). Among the custom architectures, Model V2 (91.2% accuracy, 90.1% sensitivity) performed best, narrowing the gap to transfer learning through dual-convolution blocks, batch normalization, and aggressive augmentation.

On preprocessing and augmentation, strict consistency between the training and deployment preprocessing pipelines (resizing to 224×224 , 1/255 normalization, and consistent RGB channel ordering) proved essential for preserving model performance at inference time. Aggressive augmentation reduced the train–validation gap to 1.4% for EfficientNetB0, although its specific contribution was not isolated through a formal ablation study, a limitation we acknowledge.

On deployment, the optimal model was integrated into a Flask-based web application featuring real-time inference, strict input validation, UUID-based anonymization, and automatic post-inference file deletion. Empirically measured latency and throughput confirm that the system responds within clinically acceptable timeframes on GPU-equipped hardware.

Immediate priorities for future work include: external validation on multi-institutional and cross-vendor datasets; bias and fairness auditing using demographic metadata and subgroup performance analysis; ablation studies to isolate the contribution of individual augmentations and remove anatomically implausible transformations; prospective clinical evaluation of real-world impact on diagnostic accuracy and workflow; regulatory validation, including HIPAA risk analysis, business associate agreements, audit logging, and a GDPR Data Protection Impact Assessment; explainability integration via Grad-CAM and uncertainty quantification to build clinician trust; and a multi-class extension to classify fracture type and localize fracture regions.

References

1. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
2. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
3. G. Litjens et al., "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.
4. A. Esteva et al., "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, Feb. 2017.

5. P. Rajpurkar et al., "MURA: Large dataset for abnormality detection in musculoskeletal radiographs," arXiv:1712.06957, 2017.
6. E. J. Topol, "High-performance medicine: The convergence of human and artificial intelligence," *Nat. Med.*, vol. 25, no. 1, pp. 44–56, Jan. 2019.
7. C. J. Kelly, A. Karthikesalingam, M. Suleyman, G. Corrado, and D. King, "Key challenges for delivering clinical impact with artificial intelligence," *BMC Med.*, vol. 17, no. 1, p. 195, Oct. 2019.
8. E. J. Hwang et al., "Development and validation of a deep learning-based automatic detection algorithm for rib fractures on chest radiographs," *Radiology*, vol. 293, no. 3, pp. 683–691, Dec. 2019.
9. D. S. Lindsey et al., "Deep neural networks improve radiologists' performance in fracture detection," *Proc. Natl. Acad. Sci. USA*, vol. 116, no. 46, pp. 23370–23375, Nov. 2019.
10. H. Smalley et al., "External validation of a deep learning algorithm for detection of wrist and ankle fractures on radiographs," *Emerg. Radiol.*, vol. 26, no. 5, pp. 533–541, Oct. 2019.
11. J. Cohen et al., "Artificial intelligence versus radiologists for wrist fracture detection: A multicenter diagnostic accuracy study," *Radiology*, vol. 307, no. 2, e222091, Apr. 2023.
12. H. Jung et al., "Artificial intelligence for fracture detection across imaging modalities: A systematic review and meta-analysis," *Lancet Digit. Health*, vol. 6, no. 2, pp. e89–e101, Feb. 2024.
13. M. Alshafeiy et al., "Lightweight CNN for fracture detection on edge devices in low-resource healthcare settings," *IEEE J. Biomed. Health Inform.*, vol. 28, no. 3, pp. 1456–1467, Mar. 2024.
14. M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6105–6114.
15. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.
16. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

نظام تشخيص بمساعدة الحاسوب للكشف عن كسور العظام في صور الأشعة السينية

من نماذج التعلم العميق إلى النشر السريري

المختار عبدالله المبروك¹، إبراهيم علي المراهق² [0000-0003-3759-8889]

جامعة غريان، غريان، ليبيا¹
جامعة طرابلس، كلية تقنية المعلومات، طرابلس، ليبيا²

mokhtaralmbrouk8@gmail.com

المستخلص. تُعد كسور العظام من أكثر إصابات الجهاز الهيكلي شيوعاً، إلا أن تفسير صور الأشعة السينية لا يزال عرضة للأخطاء البشرية، ولا سيما في المناطق التي تعاني من نقص أطباء الأشعة المتخصصين. تقدم هذه الورقة البحثية تطويراً ونشراً سريرياً لنظام متكامل للتشخيص بمساعدة الحاسوب (Computer-Aided Diagnosis - CAD) للكشف الآلي الثنائي عن كسور العظام.

تم إجراء تقييم منهجي لأربع معماريات مخصصة للشبكات العصبية الالتفافية (Convolutional Neural Networks - CNNs)، بالإضافة إلى نموذجين للتعلم بالنقل (Transfer Learning)، هما **EfficientNetB0** و **VGG16**، باستخدام مجموعة بيانات عامة لصور الأشعة السينية للجهاز الهيكلي (Kaggle: ahmedashrafahmed/bone-fracture)، (الإصدار 1). وللمحد من مشكلة الإفراط في التكيف (Overfitting) وتحسين متانة النماذج، تم تطبيق خط أنابيب مكثف لتعزيز البيانات (Data Augmentation) على مجموعة التدريب. أظهرت النتائج التجريبية أن التعلم بالنقل يتفوق بصورة ملحوظة على المعماريات المخصصة، حيث حقق نموذج **EfficientNetB0** أفضل أداء تشخيصي، بدقة بلغت **93.8%**، ومساحة تحت منحنى الخصائص التشغيلية للمستقبل (AUC-ROC) بلغت **0.96**، إضافة إلى حساسية سريرية بالغة الأهمية بلغت **92.7%**.

ولتحويل هذه النتائج إلى تطبيق عملي ذي فائدة سريرية، تم دمج النموذج الأمثل في تطبيق ويب آمن وخفيف الوزن قائم على إطار العمل Flask، يوفر إمكانيات الاستدلال الفوري (Real-Time Inference)، والتحقق الصارم من صحة المدخلات، وإخفاء هوية الملفات باستخدام معرفات فريدة عالمية (UUID)، فضلاً عن الحذف التلقائي للبيانات بعد عملية الاستدلال، بما يضمن الامتثال لمعايير خصوصية المرضى.

الكلمات الدلالية. التشخيص بمساعدة الحاسوب، كشف كسور العظام، التعلم العميق، التعلم بالنقل، **EfficientNetB0**، تحليل الصور الطبية.