Detection and Classification of Bone Fracture Based on Machine Learning

King Saud University | Bachelor's Graduation Project | December 2024

Team	Raed Al Shabib | Turki Aldosari | Nawaf Alfayez | Abdullah Alyahya | Rakan Hashim
Supervisor	Dr. Habib Dhahri

Project Overview

Developed an AI system to automatically detect and classify bone fractures from X-ray images, reducing diagnostic errors and supporting healthcare professionals. Two deep learning architectures — **VGG-19** and **ResNet-50** — were implemented with transfer learning on the MURA medical dataset.

Dataset — MURA

- 20,341 X-ray images across 3 bone types
- Elbow: 5,396   Hand: 6,003   Shoulder: 8,942
- Labels: Normal vs. Fractured
- Source: Kaggle (Stanford MURA v1.1)

Technologies

- Python, TensorFlow, Keras
- OpenCV, Matplotlib, Seaborn
- scikit-learn (metrics)
- VGG-19 & ResNet-50 (ImageNet weights)

Methodology

1. **Preprocessing:** Gaussian noise reduction, Adaptive Histogram Equalization (AHE), Canny edge detection, and data augmentation (rotation ±20°, flips, zoom, shear).
2. **Model Architecture:** Pre-trained base (frozen) + custom head: GlobalAveragePooling2D → Dense(512, ReLU) → Dropout(0.5) → Dense(3, Softmax). Input: 224×224 RGB.
3. **Training:** 50 epochs, batch size 32, Adam (lr=1e-4), categorical cross-entropy. Class weighting applied to handle dataset imbalance.
4. **Evaluation:** Accuracy, Precision, Recall, F1-Score, Confusion Matrix, and full classification report per class.

Implementation Code

Both models share the same data pipeline. The only difference is the base model and its preprocessing layer.

Step 1 — Library Imports (shared)

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from tensorflow.keras.applications import VGG19, ResNet50
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 2 — Dataset Preparation & Augmentation (shared)

```
# Augmentation for training only
train_datagen = ImageDataGenerator(
rotation_range=20, width_shift_range=0.2, height_shift_range=0.2,
shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen = ImageDataGenerator() # no augmentation on test
selected_classes = ["XR_ELBOW", "XR_HAND", "XR_SHOULDER"]
train_generator = train_datagen.flow_from_directory(
train_dir, target_size=(224,224), batch_size=32,
class_mode="categorical", classes=selected_classes, shuffle=True)
test_generator = test_datagen.flow_from_directory(
test_dir, target_size=(224,224), batch_size=32,
class_mode="categorical", classes=selected_classes, shuffle=False)
```

Step 3 — Class Weighting & Model Training (shared)

```
# Handle class imbalance with weighted loss
class_counts = train_generator.classes
total = len(class_counts)
class_weights = {
0: total / (3 * np.bincount(class_counts)[0]), # XR_ELBOW
1: total / (3 * np.bincount(class_counts)[1]), # XR_HAND
2: total / (3 * np.bincount(class_counts)[2]) # XR_SHOULDER
}
history = model.fit(
train_generator, epochs=50,
validation_data=test_generator,
class_weight=class_weights)
```

Step 4 — Evaluation (shared)

```
test_generator.reset()
y_pred = model.predict(test_generator).argmax(axis=1)
y_true = test_generator.classes
# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=class_names, yticklabels=class_names)
# Classification Report
print(classification_report(y_true, y_pred, target_names=class_names))
```

Model-Specific Architecture

Each model uses the same custom head. Only the base network and its preprocessing differ.

VGG-19 Model

```
# VGG-19 — frozen ImageNet weights
base = VGG19(weights="imagenet",
include_top=False,
input_shape=(224,224,3))
base.trainable = False
model = models.Sequential([
layers.Lambda(lambda x:
tf.keras.applications
.vgg19.preprocess_input(x)),
base,
layers.GlobalAveragePooling2D(),
layers.Dense(512, activation="relu"),
layers.Dropout(0.5),
layers.Dense(3,
activation="softmax")
])
model.compile(
optimizer=tf.keras.optimizers
.Adam(lr=1e-4),
loss="categorical_crossentropy",
metrics=["accuracy", F1Score(),
Precision(), Recall()])
```

ResNet-50 Model

```
# ResNet-50 — frozen ImageNet weights
base = ResNet50(weights="imagenet",
include_top=False,
input_shape=(224,224,3))
base.trainable = False
model = models.Sequential([
layers.Lambda(lambda x:
tf.keras.applications
.resnet50.preprocess_input(x)),
base,
layers.GlobalAveragePooling2D(),
layers.Dense(512, activation="relu"),
layers.Dropout(0.5),
layers.Dense(3,
activation="softmax")
])
model.compile(
optimizer=tf.keras.optimizers
.Adam(lr=1e-4),
loss="categorical_crossentropy",
metrics=["accuracy", F1Score(),
Precision(), Recall()])
```

Results

Metric	Class	ResNet-50	VGG-19
Precision	XR-Elbow	99%	100%
	XR-Hand	100%	100%
	XR-Shoulder	100%	99%
Recall	XR-Elbow	100%	99%
	XR-Hand	99%	100%
	XR-Shoulder	100%	100%
Overall Accuracy	All classes	**~100%**	**~100%**

Key Findings

- Both models achieved near-perfect classification accuracy on all three bone types.
- ResNet-50 excelled on shoulder X-rays (563/563 perfect); VGG-19 excelled on hand X-rays (460/460 perfect).
- Class weighting effectively handled dataset imbalance during training.
- Transfer learning from ImageNet significantly reduced training time while maintaining high accuracy.
- ResNet-50 showed faster loss convergence, suggesting more efficient training dynamics.