浙江工业大学

本科留学生毕业设计说明书（论文）

Undergraduate International Students' Graduation Project Report (Thesis)

题目：尺桡骨骨折识别的设计与现

TITLE：THE DESIGN AND IMPLEMENTATION OF DISTAL RADIUS FRACTURES RECOGNITION FROM HAND-WRIST RADIOGRAPHS

姓名 Name of student：Moussaid Aicha

学号 Student ID: L201726630123

年级 Grade：2017

护照号 Passport Number: YE6374756

指导教师 Project Supervisor: Hao Pengyi

专业班级 Major&Class: Software Engineering

所在学院 College of Computer Science and Technology

提交日期 **Date: 2021** 年 **6**

Abstract

DRFs (distal radius fractures) are the most common upper-extremity fractures in humans. As a result, they contribute for a large percentage of the conditions seen in emergency rooms and hospitals around the world. Hand-wrist X-Ray images are considered as a crucial tool in the diagnosis of DRFs, which is why the implementation of Computer-Aided Detection and Diagnosis systems (CAD) that thoroughly analyze these radiographs with reliable accuracies, is pressing and essential.

In this thesis, an automated and efficient diagnostic approach combining deep learning and mobile app development is used to create a reliable auxiliary system that is reliable and efficient for doctors and medics. It mainly includes three parts: (i) We explore the classification of left hand and right hand, and the classification of front hand and side hand, which is useful in automatically write a summary in the diagnosis systems. (ii) We perform DRF detection and classification, several methods are compared and analyzed. The best one is selected for using in the diagnosis system. (iii) We develop a mobile application for an accessible and fast portable DRF diagnosis. Our project successfully reached the expected results with promising scope of improvement.

Keywords: Computer-aided Detection and Diagnosis (CAD), Distal Radius Fractures detection, Hand-wrist Radiographs, Mobile App Development

摘要

桡骨远端骨折是人类最常见的上肢骨折之一。因此，在世界各地的急诊室和医院中，他们造成了很大比例的疾病。手腕部 X 射线图像被认为是桡骨远端骨折诊断的重要工具，因此实现计算机辅助检测和诊断系统（CAD）对这些手腕部 X 光片进行全面可靠的分析是迫切和必要的。

本文采用深度学习，开发自动化、高效的移动应用程序用于桡骨远端骨折的诊断，为医务人员创建一个可靠、高效的辅助系统。本文的主要工作包括三个部分：（1）本文探讨了手腕部 X 光片中的左手和右手的自动分类，以及正面手腕和侧面手腕的自动分类，这有助于在诊断系统中自动编写总结报告；(ii）本文对桡骨远端骨折进行了骨折检测和骨折类别的识别，并对几种方法进行了比较和分析，选择最佳的模型应用于诊断系统中； (iii）本文基于对左右手、正侧面手的识别和对骨折的检测与骨折类型的识别，开发了一个移动应用程序，用于快速准确的桡骨远端骨折诊断。该应用达到了预期的效果，但还有很大的改进空间。由于对桡骨骨折的 B 型和 C 型的识别不够理想，下一步，本文将对桡骨骨折的类型识别问题进行更加深入的研究，使得桡骨骨折类型的识别更加精准。

关键词：计算机辅助设计、桡骨骨折检测、手腕部 X 光片、移动应用开发

Contents

Chapter 1. Introduction **1**

1.1 Research Background 1

1.2 Research Status 2

1.3 Research Goals and Significance 3

1.4 Organization of the Thesis 4

2. Chapter 2. Related Works **5**

2.1 The Traditional DRF Diagnostic Approaches 5

2.2 The ML DRF Diagnostic Approaches 5

2.2.1 Convolutional Neural Network 6

2.2.2 Applications in Radiology 7

3. Chapter 3. Methods, Experiments and Evaluation **8**

3.1 Overview 8

3.2 The architecture of Used CNNs 9

3.3 Deep Learning 16

3.3.1 Orientation Classification 16

3.3.2 DRF Detection 16

3.3.3 DRF Classification 17

3.4 Experiments 18

3.4.1 Dataset 18

3.4.2 Models Training 20

3.5 Results 20

3.5.1 Orientation Classification 20

3.5.2 DRF Detection 22

3.5.3 DRF Classification 25

3.6 Dataset Analysis 27

4. Chapter 4. System Design and Implementation 29

4.1 Overview 29

4.2 Basic Information and System Implementation 29

4.2.1 Language 29

4.2.2 Main Concept 30

4.2.3 Models Deployment 31

4.2.3 Database 33

4.3 DRF Diagnosis App Interface 34

4.4 Scope of Improvement 39

5. Chapter 5. Conclusion 40

5.1 Summary... 40

5.2 Future Works 40

6. References.. 41

7. Acknowledgement 44

Chapter 1. Introduction

1.1 Research Background

Our modern world is converging towards the exclusion of manual labor, automatizing tasks and minimizing the error rate to the max in various fields. That is why the importance of Artificial Intelligence (AI) and its breakthroughs in all existing fields is making the worldwide spending on its development to double, reaching by the year 2024 a total of $110 Billion, according to the report [1]. While the achievements AI has accomplished in various domains are revolutionary and worth discussing, our focus falls on the healthcare sector, where the development of AI-based Computer Aided Diagnosis and Detection (CAD) systems is crucial.

Hospitals around the world suffer from the lack of reliable systems that automatize repetitive tasks such as examining radiographs, which puts the doctors under unnecessary straining burden, and the patients to face slow, and sometimes, inaccurate diagnosis. For instance, diagnosing and detecting Distal Radius Fractures (DRF) relies primarily on conventional radiographs as cited by Waever D et al. in their research [2], and non-orthopedic surgeons or young radiologists at emergency departments are swarmed with high amounts of radiographs assessment. Therefore, implementing a reliable, efficient assistant technology can be revolutionary.

Artificial Intelligence (AI) has succeeded in making remarkable achievements and progress in image processing and interpretation [3, 4]. Deep learning, which is a branch of AI, has rapidly become a powerful method in image analysis ever since 2012, with the use of CNNs which are suitable for analyzing image features [5, 6]. With the definite increase in numbers of experimental trials that apply deep learning in medical image analysis in different branches, including the automated analysis of lung nodules [7], and traumatic orthopedics [8, 9, 10], AI has proven its ability in matching and even surpassing human abilities with the excellent performance that the CNNs proved to achieve, as shown in these studies [11, 12].

1.2 Research Status

In the field of medical imaging, including computer-aided diagnosis (CAD), radiomics, and medical image processing, the use of machine learning (ML) is increasingly growing. Deep learning (DL), a form of machine learning that first appeared in the field of computer vision and has since grown in popularity across a wide range of fields, has become quite popular. It all began in late 2012, when a deep-learning approach based on a convolutional neural network (CNN) won the most prestigious worldwide computer vision competition, ImageNet Classification. Since then, researchers from a wide range of fields, including medical imaging, have been active participants in the rapidly expanding area of deep learning.

Prior to the advent of deep learning, machine learning with feature feedback (or feature-based machine learning) was the standard. This led ML to benefit from the revolutionary strength of deep learning, as it allows for direct learning of image data without the need for object segmentation or feature extraction, as further clarified by Suzuki, K. [13]. The introduction of Artificial Neural Networks (ANN), which were modeled after the multilayered human cognition system, the development in their architectures and their combination with big healthcare data, resulted in the current popularity that deep learning now has.

The use of deep learning and AI in radiology is still in its early stages. A strong shared understanding of the technology, as well as the most suitable method of radiology practice and workflow by both radiologists and computer scientists/engineers, is one of the most important factors for the advancement of AI and its proper clinical acceptance in radiology. As discussed by this article of Korean J Radiol. [14], wide, completely annotated databases are needed to advance AI creation in medical imaging, thanks to ImageNet's recent technological advances. This will be critical for both preparing and evaluating the deep learning network. Many radiologists must be actively involved in the development of a broad medical imaging database, and by consequence, allow better CAD to flourish.

1.3 Research Goals and Significance

The overall goal of the study is to compare the performance and summarize the deep learning methods of Distal Radius Fracture detection and classification from hand-wrist X-ray images with a focus on object detection and multi-label classification. The study conducts double classification to determine the nature of the examined radiograph (Left/Right hand, AP/Lat view), then it works on detecting the fracture present on it, to finally determine its type with another set of classification (Fracture Type A, B, C). The research then converts the saved PyTorch and Keras models in an appropriate format that can be deployed in reliable platforms that help the target users, doctors and medics, perform an efficient automated DRF diagnosis.

The system's core functionality is the auxiliary diagnosis. which will help the users in the medical field to easily and quickly diagnose and detect the DRFs from the hand-wrist radiographs, by employing different CNNs, and outputting the optimal conclusion and result. Even if the approaches to this topic using different CNNs who proved to have very reliable accuracies are widely available, the real challenge falls on deploying these models successfully in reliable platforms.

The system should meet the functional requirements above as well as non-functional requirements such as usability and flexibility. The analysis of both will be discussed in the requirement section of the paper. The main significances of this work are summarized as follows:

1. Successful training, testing and evaluation of different CNNs used regarding various factors (Dataset, Time, Accuracy).
2. Comparison of the CNNs performance and deploying the most optimal model in the designed platform.
3. Development of a system with appropriate UX and a user-friendly UI and providing the doctors with a reliable auxiliary diagnosis system.

1.4 Organization of the Thesis

The thesis at hand will begin with a presentation of the existing Distal Radius Fractures diagnosis approaches, namely the traditional one performed by physicians in hospitals, and the modern one employing the power of deep learning and the application of CNNs in fracture radiology. It will briefly explain CNNs as they are an important factor in this thesis and cite the important achievements in the recent work related to the viewpoint of this project. In Chapter 3 methods, experiments, and evaluation, we present the work's employed approaches, experiments, comparison, and discussion. The details of the dataset will also be introduced, with an explanation of the data augmentation performed on it. In Chapter 4 System design, the lengthy procedure of the development of our diagnosis system and the requirements analysis will be laid out in details. The 5th chapter discussion and limitations will discuss the struggles met during the overall achievement of the project, the limitations of this study as well as future work. Finally, concluding commentary is presented in the conclusion, followed by references and acknowledgement.

Chapter 2. Related Works

2.1 The traditional DRF diagnostic approaches

The traditional approach in diagnosing Distal Radius Fractures is a three-step process, that starts with consulting a physician. The latter gathers information from the patient about the symptoms, how they started and what triggers them. The second step consists of the physical exam, where patients' wrists and forearms are examined for any discoloration or dislocation, and where they are asked by their physician, to perform certain hand or wrist movements to gauge if there is any pain or difficulty involved. The last step of this process and the most reliable one, is the imaging step. It is almost always required to get an X-Ray exam of the wrist from several angles, and sometimes other imaging options are also ordered under certain circumstances:

- MRI/CT tests for examining complex fractures and revealing additional injuries to the nearby soft tissues. They are performed by injecting contrast dye (also called MRI/CT-arthrograms) to help show the damage in the ligaments or other soft tissues.
- Bone Scan (Scintigraphy) tests for detecting metabolic changes in the bone from the fractures, which is helpful in hard-to-see bone structures.

Imaging also reveals the proper classification of the fracture, which is necessary to taking the right treatment approach, allowing physicians to determine which fractures are susceptible to orthopedic treatment, and which require surgery. This type of fractures' diagnosis often requires X-Rays of the wrist in two positions, namely anterior-posterior and lateral. The treatment's decision-making takes into account additional factors like the age and activity level of the patient.

2.2 The ML DRF diagnostic approaches

As previously discussed in the Chapter 1, ML has played a spinning role in many fields, and importantly healthcare. By benefiting from deep learning techniques and improving with their development, various works flourished in medical imaging and came up with reliable results. This happened by using Convolutional Neural Networks (CNN) and applying them in computer vision accordingly.

2.2.1 Convolutional Neural Networks

It is very important to understand the concept of CNNs and their inner working for it is an essential base in this thesis. A Convolutional Neural Network is one of the most impressive forms of Artificial Neural Networks (ANN) architecture as it is also most popularly used for analyzing images. Their design was inspired by the organization of the Visual Cortex and is analogous to the architecture of the connectivity pattern of Neurons in the Human brain. We can think of a CNN to be an ANN that has some type of specialization for being able to pick out or detect patterns and make sense of them. This pattern detection is what makes a CNN useful for image analysis. Each neuron in a CNN is responsible for the data processing in its receptive field only, while the layers it contains are organized in a way that allows them to first, detect simple patterns (i.e., lines, curves, etc.), then move to patterns of higher complexity (i.e., faces, objects, etc.) which enables sight to computers.

Typically, a CNN contains three layers: a convolutional layer, a pooling layer, and a fully connected layer.

- **Convolutional Layer**: The CNN's backbone is the convolutional layer, which is made up of independent filters that are randomly initialized and searches for certain features in an image. These filters provide the parameters that are learned throughout the network. The dot products of the filters in the previous convolution layers are used in the deeper convolution layers [15].

- **Pooling Layer**: The pooling layer uses a summary statistic of neighboring outputs to replace the network's output at specific spots. This reduces the representation's spatial size, which reduces the amount of computation and weights necessary. Every slice of the representation is individually handled throughout the pooling operation. Because of the translation invariance provided by pooling, an item may be recognized regardless of where it appears on the frame.

- **Fully Connected Layer**: The FC layer maps the representation between the input and output. It uses high-level filtered features to produce the result based on the class with the highest probability [15, 16].

2.2.2 Applications in Radiography

The application of CNNs and their feasibility in the detection of fractures from radiographs was investigated in previous studies, which led to promising results that align with the goal of the thesis in hand. Inception-V3 was trained to recognize wrist fractures on lateral wrist radiographs and achieved an AUC up to 0.95, a 0.9 sensitivity and a 0.88 specificity, as inducted by Kim and MacKinnon [17]. Olczak et al. [18] on the other hand, led a study to detect fractures on hand, wrist, and ankle radiographs, employing a VGG-16 network and an accuracy of 83%, which matched that of radiologists (who had an 82% accuracy). Chung et al. [19] attempted in their research to detect and classify proximal humerus fractures using shoulders radiographs, where their ResNet showed superior top-1 accuracy of 96%, surpassing that of orthopedists (93%).

Another example of a CNN that exceeded the orthopedists' accuracy, achieving 96% and 92% respectively, is the VGG-16 employed in the study conducted by Urakawa et al. [20]. All the mentioned studies prepared their training and testing datasets by manually cropping the images in specific matrix sizes before inputting them in the CNNs. With this uniformity, the recognition of the Regions of Interest (ROIs) on the images was faster and more accurate, thereby outstandingly improving the efficiency of the CNNs during learning and testing.

The conducted study of Kaifeng et al. [21] employed a Faster R-CNN model to automatize the annotation of the ROIs on images, which came as a successful substitution to manual cropping (Urakawa et al. [20]). This resulted in a substantial decrease in both bias and processing time. They saw that combining an Inception-v4 with a faster R-CNN to detect DRFs for the wrist radiographs used could have great potential since these images are presented with both ROIs and irrelevant regions. Their view was proven when the trained Faster R-CNN achieved a 100% success rate in the automatic ROIs annotation which formed a reliable auxiliary algorithm to the Inception-v4, which was trained on distinguishing images with DRFs from normal images. This led to a diagnostic accuracy similar to that of orthopedists and superior to that of radiologists.

Chapter 3. Methods, Experiments and Evaluation

3.1 Overview

The original radiograph is initially subjected to the deep learning and image processing pipeline which consists of three main parts as shown in Fig. 3-1. The first step is the orientation classification part, which detects the nature of the radiograph (Left/Right hand, AP/Lat view). The second part is the ROI detection using a fast detection algorithm, to extract an outlined picture highlighting the fracture. The extracted image with the highlighted ROI will then be subjected to a DRF classification to determine its type (Type A, Type B, Type C). This will mark the end of the deep learning part of this project.

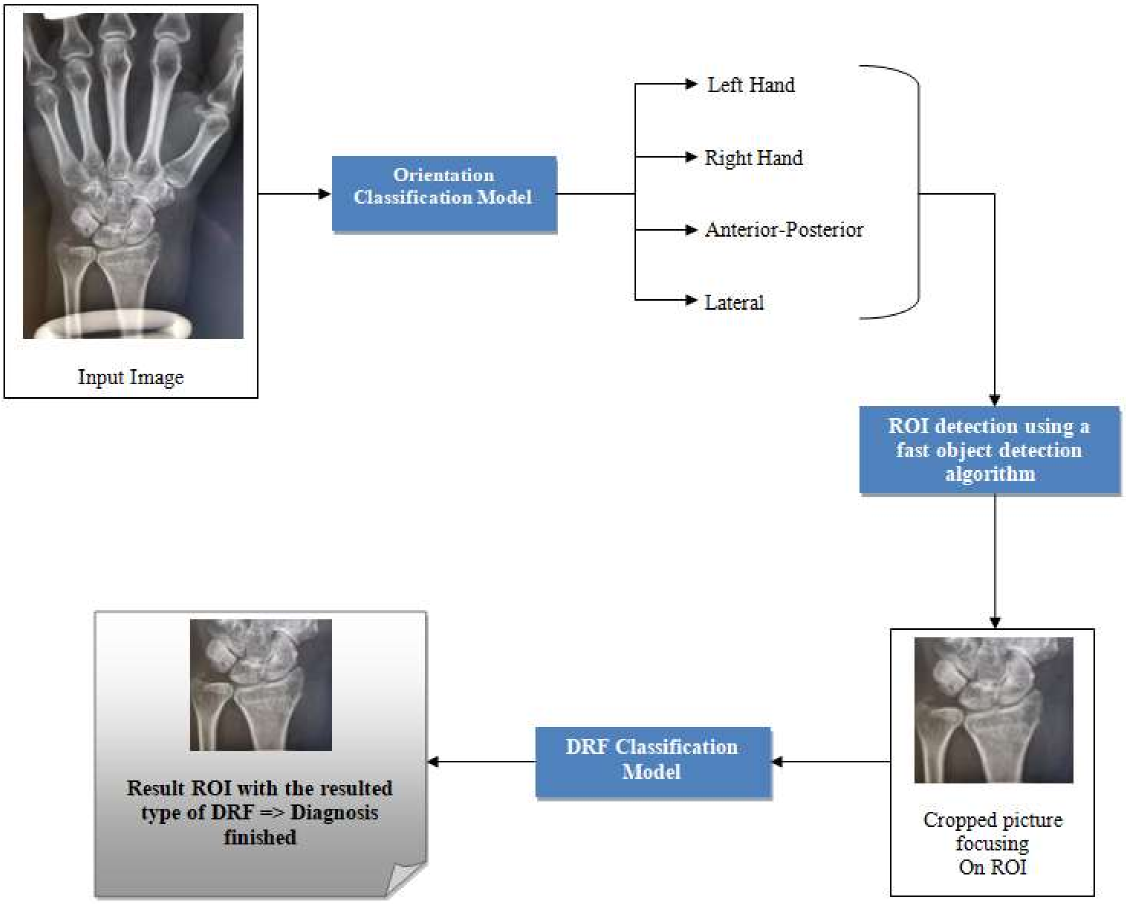

Input Image
Orientation Classification Model
Left Hand
Right Hand
Anterior-Posterior
Lateral
ROI detection using a fast object detection algorithm
Cropped picture focusing On ROI
DRF Classification Model
Result ROI with the resulted type of DRF => Diagnosis finished

Fig. 3-1. Explanatory flowchart of the design and implementation of distal radius fractures from handwrist radiographs

3.2 The architecture of Used CNNs

This part of the thesis is responsible for the theoretical explanation and details about the CNN architectures used in implementing and testing our models, throughout the three deep learning parts of the project.

- **ResNet-50**
  The ResNet-50 belongs to the ResNet model variants, containing 48 Convolutional layers, 1 MaxPool, and 1 Average Pool layer, and has over 23 million trainable parameters. The importance of this network is that it revolutionized the concept of training ultra-deep neural networks while still achieving great performance. Deep convolutional neural networks are very important for their ability to identify low, mid, and high-level features from images, allowing the achievement of very high accuracies. Although, this was not an easy concept to achieve. There was the problem of vanishing and exploding gradients who were solved using many approaches that worked on CNNs with tens of layers and making them converge. But with this convergence came the problem of accuracy saturation then degradation, without any overfitting involved. This was rectified by two models, one with 20 layers, and the other was constructed in the exact same way, while only adding identity layers to it. This produced higher training errors which should not have happened since the added layers were only identity ones, as illustrated in Fig.3-2.

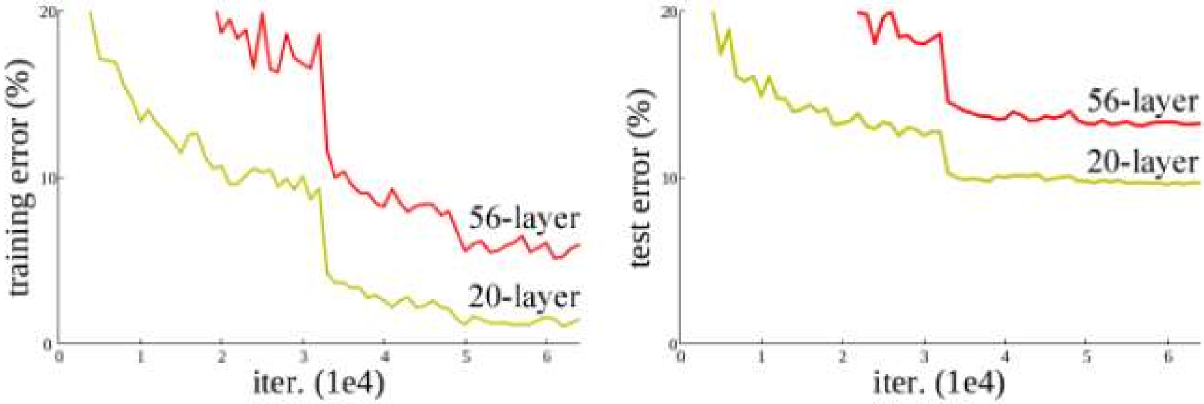


Fig. 3-2 Illustration of the shallow model vs deep model approach

The authors finally addressed this problem accordingly by introducing deep residual networks, equipped with shortcut connections that simply perform identity mappings. They expressly allowed the layers to fit a residual mapping and annotated it as H(x), and they explicitly allowed the nonlinear layers to fit another mapping F(x): =H(x)x, resulting in H(x): =F(x)+x, as shown in Fig. 3-3.

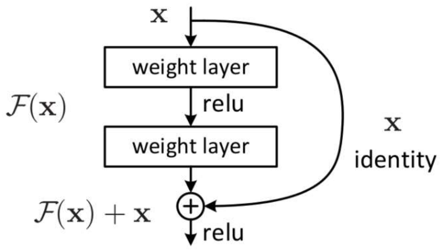
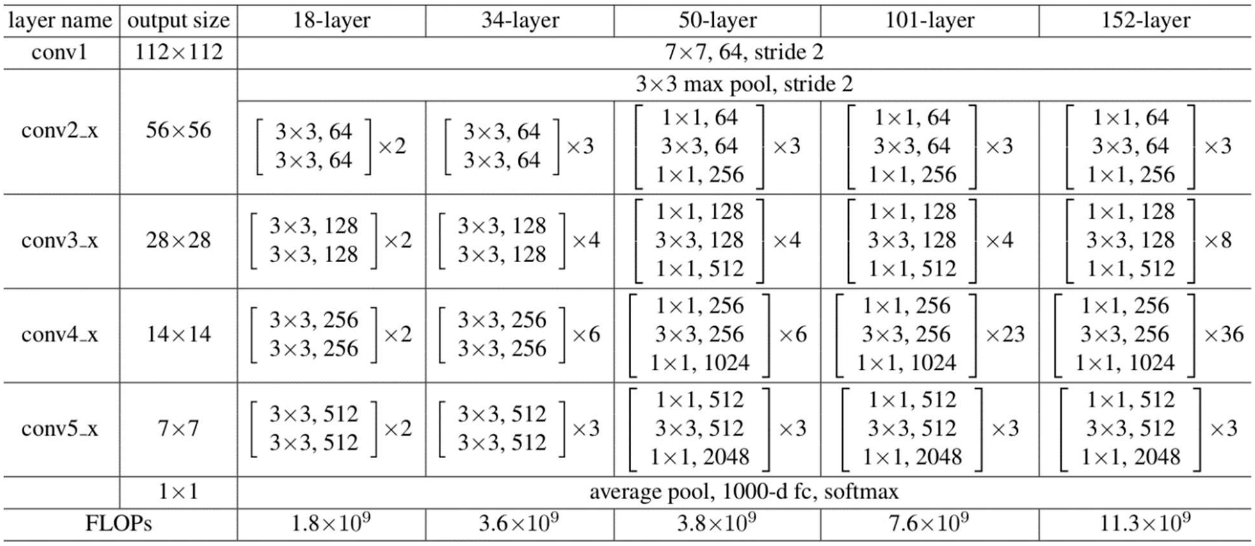


Fig. 3-3 Residual Mapping

This resulted in no additional parameters added to the model and the computational time stayed optimal. Table 3-1 further summarizes the different types of ResNet according to the number of layers.

Table 3-1 The Architecture of ResNet50



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

- **VGG-16**

  Developed by K. Simonyan and A. Zisserman, VGG-16 model was introduced as an improved AlexNet by replacing large kernel-sized filters with multiple 3x3 kernel-sized filters successively one after the other. In 2014, it was utilized to win the ImageNet (ILSVR) competition. It is still regarded as an outstanding vision model, despite the fact that more recent breakthroughs like as Inception and ResNet have outperformed it.

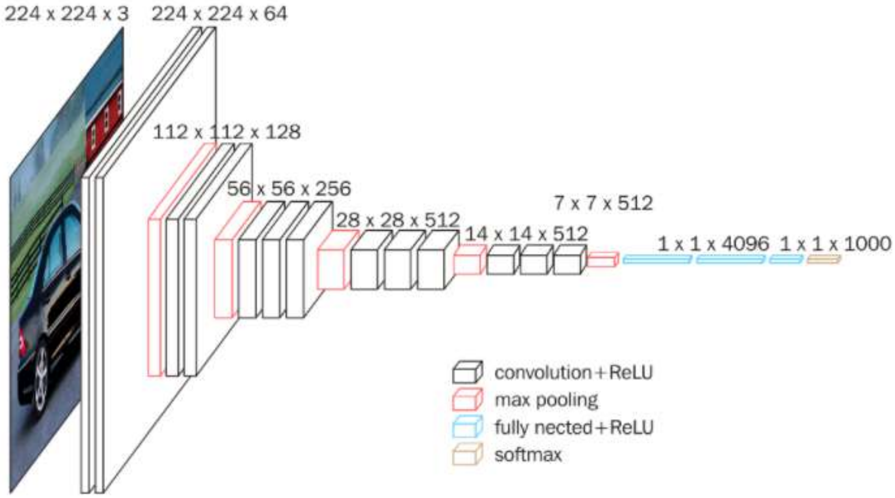
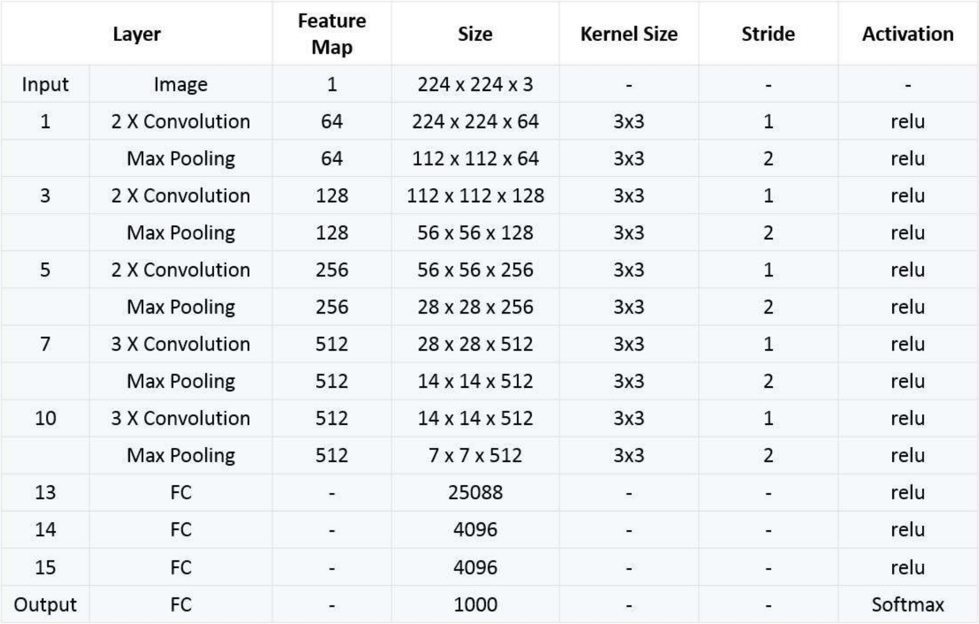


Fig. 3-4 The VGG-16 Model on ImageNet [28]

The size of the input to this model is fixed at 224*224*3. As input, we have a tensor of (224,224,3). This input contains three channels (R,G,B) with a pixel size of 224*224. The image is then passed through multiple convolutional layers. Throughout this passage illustrated in Fig.3-4, the size of the input image goes from 112*112*64, to 56*56*128, to 28*28*256, to 14*14*512, to reach 7*7*512. Finally, the 3 fully connected layers take the flattened features which is turned into a vector of (1, 4096), then lastly a softmax layer. This is further summarized in Table 3-2.

This makes up for an easy to implement deep learning image classification model for learning purpose despite the possible slowness and the memory resources taken.

Table 3-2 The Architecture of VGG16



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

- **Our CNN**

This model does not belong to a special CNN architecture like VGG-16 or ResNet-50, it was worth trying a simple design of a CNN and test their ability in achieving the classification needed in this part.

The model has 3 convolutional layers followed by 3 max-pooling layers. Then we find the Flatten layer to finally reach the two last dense layers.

The first layer contains 16 kernels with a size of 3. Once the convolution of the image is done, it will be passed through ReLU activation for non-linearity obtention.

Overall, 3 convolution and max-pooling layers are part of this network and are used to extract features of images. Since there were no complex features to be learned, these layers were sufficient, and no further depth was required.

The output from the previous max-pooling layer is then taken and converted into a 1D array by the Flatten layer so that it can be feed into the Dense layer. This type of layer is a regular layer of neurons in a neural network, and it is here where the learning process happens with a weight adjustment.

In our case, the CNN has two Dense layers with a number of neurons in the output layer according to the number of classes given.

- **Faster R-CNN**

Faster R-CNN, the most extensively utilized state-of-the-art variant of the R-CNN family, was originally released in 2015 [22]. The input image is delivered into the backbone convolutional neural network to begin the region proposal network (RPN). The supplied image is first adjusted so that the shortest side is 600 pixels long and the longest side is not more than 1000 pixels long.

Depending on the stride of the backbone network, the output features (represented by H x W) are frequently significantly smaller than the input image.

The network stride is 16 for both of the available backbone networks utilized in the article (VGG and ZF-Net). This indicates that two pixels in the backbone output features correspond to two spots in the input image that are 16 pixels apart. The network must learn whether an object is present in the input image at its associated position and estimate its size for each point in the output feature map. This is accomplished by placing a set of "Anchors" on the input image for each position on the backbone network's output feature map. These anchors suggest objects of various sizes and aspect ratios that could be found at this location, and this is further illustrated in Fig.3-5.

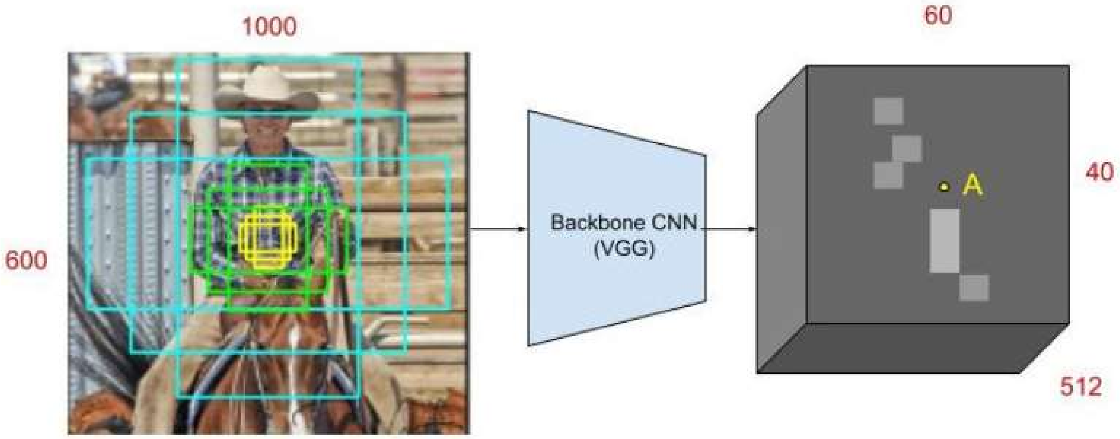
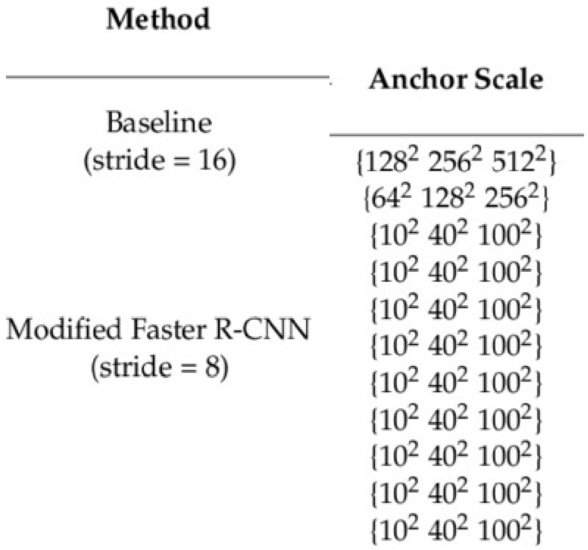


Fig. 3-5 The possible anchors in the input image in a location corresponding to point A in the feature map. [29]

As the network goes through each pixel in the output feature map, it must check whether the k related anchors spanning the input image genuinely contain objects and refine the coordinates of these anchors to produce bounding boxes as "Object suggestions" or regions of interest. On create a 512-d feature map for each site, a 3 x 3 convolution with 512 units is applied to the backbone feature map. Following that are two sister layers: an 18-unit 1 x 1 convolution layer for object classification and a 36-unit 1 x 1 convolution layer for bounding box regression. The classification branch's 18 units produce a size output (H, W, 18). This output is used to calculate the odds of each point in the backbone feature map (size: H x W) containing an object within all 9 anchors at that location. The regression branch's 36 units produce a sizeable output (H, W, 36). The 4 regression coefficients of each of the 9 anchors for each point in the backbone feature map are calculated using this output (size: H x W). These regression coefficients are used to improve the anchors that store objects' coordinates.

Table 3-3 The Architecture of Faster R-CNN



Method	Anchor Scale
Baseline (stride = 16)	$\{128^2\ 256^2\ 512^2\}$
Modified Faster R-CNN (stride = 8)	$\{64^2\ 128^2\ 256^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$
	$\{10^2\ 40^2\ 100^2\}$

- **YOLO**

Prior detection methods perform detection by repurposing classifiers or localizers. They apply the model to an image at different scales and places. The image's high-scoring sections are referred to as detections. In Yolo models, a different approach is used as a single neural network is applied to the full image. The image is divided into regions by this network, which predicts bounding boxes and probabilities for each region.

The projected probabilities are used to weight these bounding boxes. Compared to classifier-based systems, this model has significant advantages. At test time, it examines the entire image, thus its predictions are influenced by the image's overall context. In contrast to systems like R-CNN, which require thousands of network evaluations for a single image, it provides predictions with just one. This makes it a thousand times quicker than R-CNN and a hundred times quicker than Fast R-CNN [23].

Both YOLO-v3 and YOLO-v4 are used. The backbone of YOLOv4 is made up of CSPDarknet53, a spatial pyramid pooling extra module, a PANet path-aggregation neck, and a YOLOv3 head. CSPDarknet53 is a new backbone that can help CNN learn more effectively. Over CSPDarknet53, the spatial pyramid pooling block is used to expand the receptive field and filter out the most important context data. Instead of using Feature Pyramid Networks (FPN) for object identification, PANet is used for parameter aggregation at multiple detector levels in YOLOv3. This increased the Average precision and FPS by 10% and 12% compared to YOLO-v3 [24].

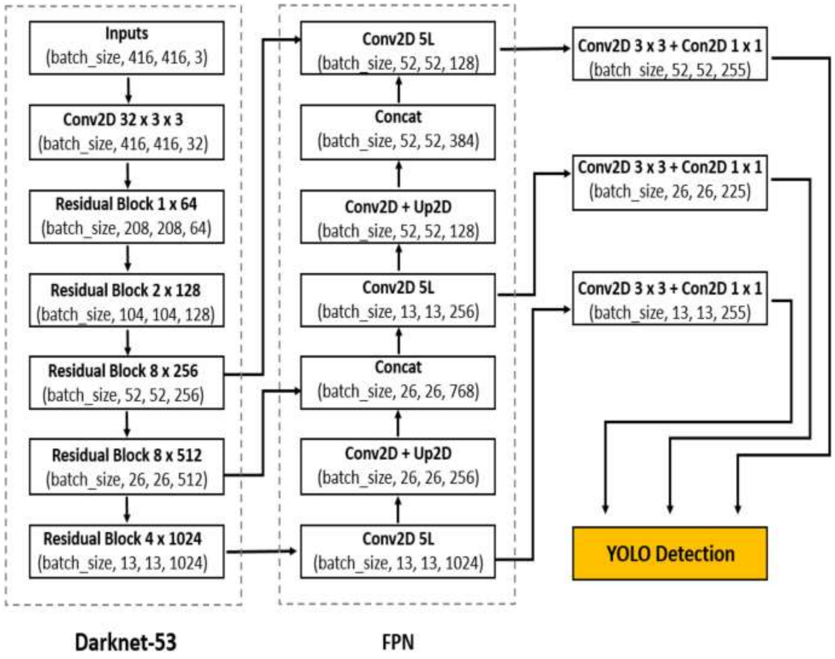

Inputs
(batch_size, 416, 416, 3)
Conv2D 32 x 3 x 3
(batch_size, 416, 416, 32)
Residual Block 1 x 64
(batch_size, 208, 208, 64)
Residual Block 2 x 128
(batch_size, 104, 104, 128)
Residual Block 8 x 256
(batch_size, 52, 52, 256)
Residual Block 8 x 512
(batch_size, 26, 26, 512)
Residual Block 4 x 1024
(batch_size, 13, 13, 1024)
Conv2D 5L
(batch_size, 52, 52, 128)
Concat
(batch_size, 52, 52, 384)
Conv2D + Up2D
(batch_size, 52, 52, 128)
Conv2D 5L
(batch_size, 13, 13, 256)
Concat
(batch_size, 26, 26, 768)
Conv2D + Up2D
(batch_size, 26, 26, 256)
Conv2D 5L
(batch_size, 13, 13, 1024)
Conv2D 3 x 3 + Con2D 1 x 1
(batch_size, 52, 52, 255)
Conv2D 3 x 3 + Con2D 1 x 1
(batch_size, 26, 26, 225)
Conv2D 3 x 3 + Con2D 1 x 1
(batch_size, 13, 13, 255)
YOLO Detection
Darknet-53
FPN

Fig. 3-6 YOLO Architecture

3.3 Deep Learning

3.3.1 Orientation Classification

This initial step of our deep learning part consists of studying and classifying the radiographs according to the nature of their view and their content. In other words, this classification gives the orientation of the radiograph for possible future management needs when connected to the platforms. The summary of this part can be illustrated in Fig. 3-7:

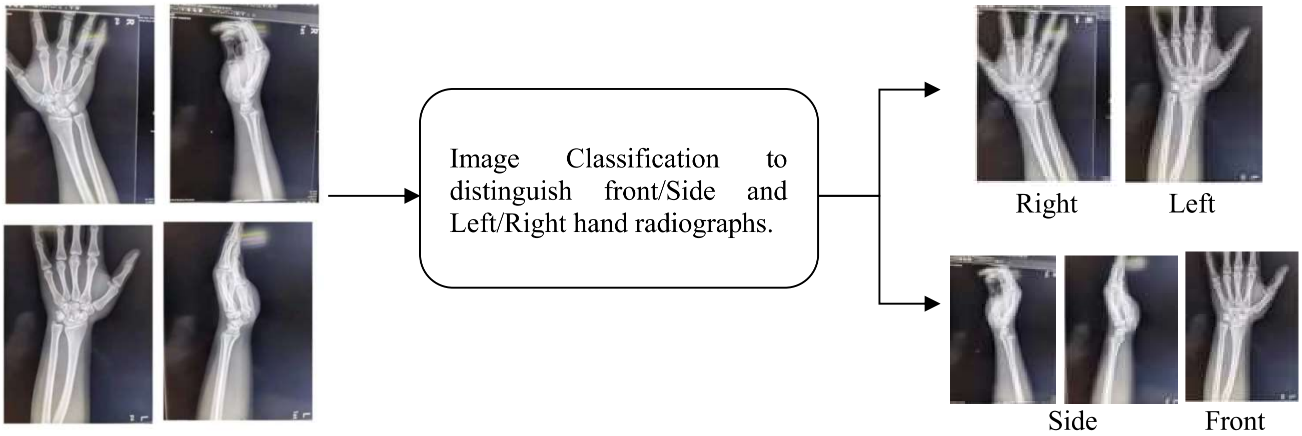


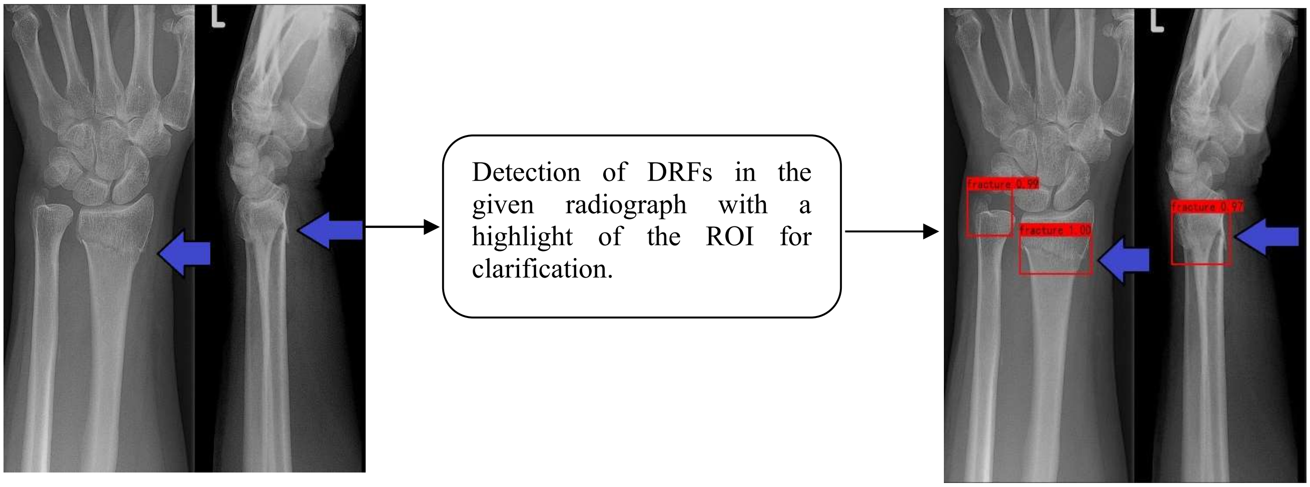
Image Classification to distinguish front/Side and Left/Right hand radiographs.
Right
Left
Side
Front

Fig. 3-7 Orientation Classification Workflow

To achieve a successful implementation of the following part, three main CNN architectures were used to test and compare their performance. We used a ResNet-50 model, a VGG-16 model, and our simply built CNN model. During this step, PyTorch, Keras and Tensorflow were used to test and find the most convenient implementation according to the nature of the repartition of the dataset. This will be further discussed in the Dataset part of the thesis.

3.3.2 DRF Detection

This part of the process is an essential one as it is meant to interact with the radiograph closely, detecting the fracture or fractures present in it. By annotating the Regions of Interest (ROIs), it highlights the suspected fracture in the input radiograph, allowing for a clearer view of the fracture's location. This is summarized in the workflow in Fig. 3-8, using real results of our trained models.



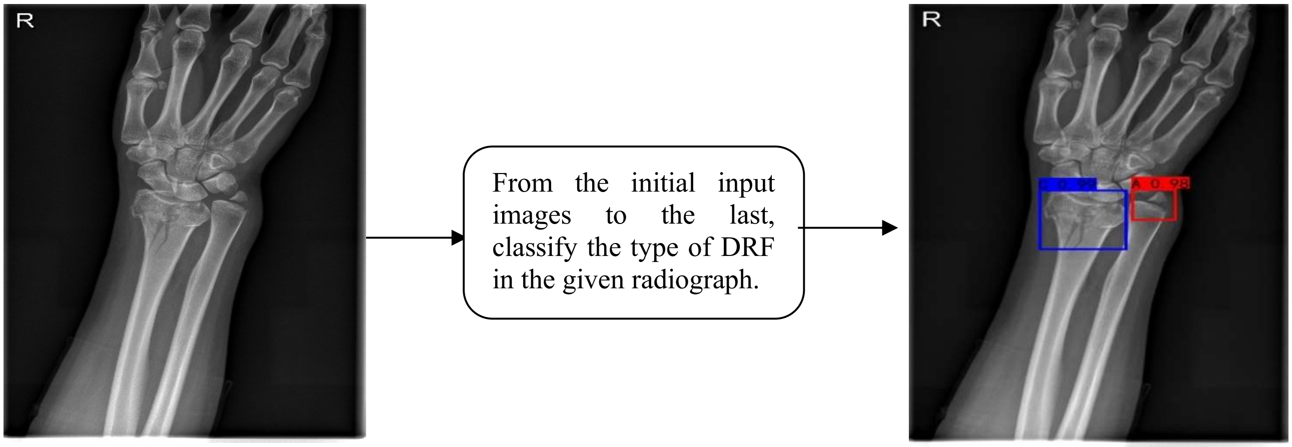
Detection of DRFs in the given radiograph with a highlight of the ROI for clarification.

Fig. 3-8 DRF Detection Workflow

For this type of object detection, Faster R-CNN, YOLO-v3 and YOLO-v4 were used to achieve different implementations and test their performance in annotating the correct ROIs of the given radiographs. The dataset in this part did not have any particular changes or adjustments.

3.3.3 DRF Classification

The last part of the deep learning process of this project falls on the correct classification of the distal radius fractures detected (Type A, Type B, Type C). It is important to classify the DRFs accordingly to make it easier for the doctors to decide the correct treatment approach. Fig. 3-9 shows the workflow describing the process of this part.



From the initial input images to the last, classify the type of DRF in the given radiograph.

Fig. 3-9 DRF Classification Workflow

For this step, the CNNs used were Faster R-CNN, YOLO-v3, YOLO-v4 and with an adjustment of the dataset given, we performed a similar classification to the orientation part, and applied the self-built CNN on it.

3.4 Experiments

3.4.1 Dataset

In this project we use 3 different types of datasets which are used for each part of the deep learning procedure accordingly. The total of the images is made of a total of 3600 picture. All the images are in 1024x1024 resolution. The first dataset was reparted into two categories and used twice. The first time to classify left/right hands, and the second time is to classify AP/Lat views. The second and third dataset used were manually annotated using LabelImg. It is a Python-based, free, open-source graphical image annotation application for naming objects in images. Annotations can be saved in PASCAL VOC format as XML files. An additional dataset was created for the third part to try the self-built CNN on it, reparting the images into Type A, B, and C.

The distribution of the images varied in the first part as it was necessary to try different distributions and gauge their impact on the classification. For example, initially, we had Left/Right repartition, AP/Lat Repartition, then we created a new repartition consisting of Right AP, Right Lat, Left AP, and Left Lat. This is all summarized in Fig. 3-10 below, with samples in Fig. 3-11.

The dataset was also partitioned into a training, testing and validation sets, with respective percentages of 80%, 10%, 10%.

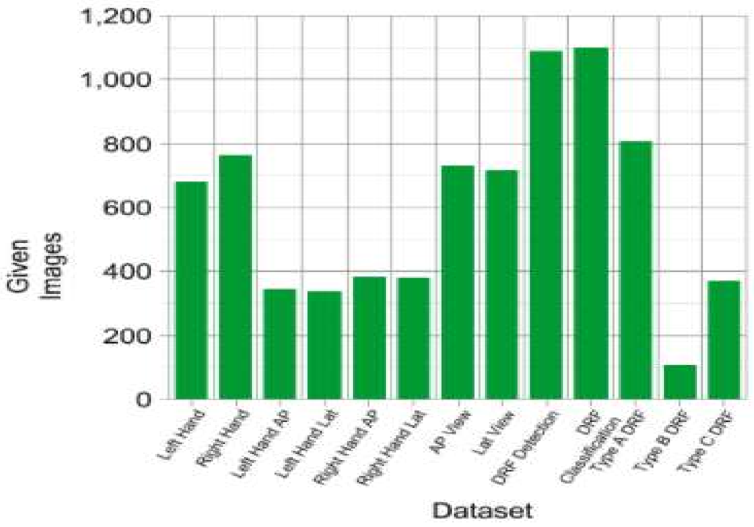


Fig. 3-10 Summary of Datasets used in the project.

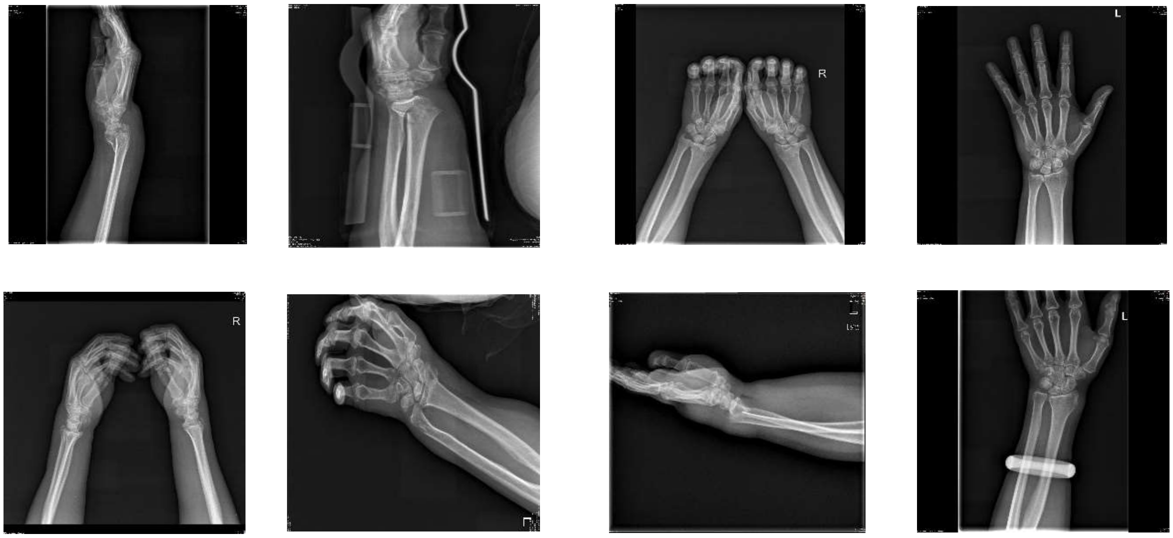


Fig. 3-11 Summary of Datasets used in the project.

To achieve great accuracy, deep learning techniques necessitate a large amount of data. Medical images, on the other hand, are difficult to obtain due to patient privacy concerns, and image annotation necessitates a strenuous and time-consuming effort by a highly skilled specialist. Data Augmentation was used on the images by manipulating the rotation, the randomFlip (Vertical) and the zoom. This improved the performance which makes it a necessity in image preprocessing.

3.4.2 Models Training

Throughout this project, PyTorch, Keras and TensorFlow were used as deep learning libraries to train the convolutional neural networks, with different versions depending on the required implementation. The training and the testing were performed on an NVIDIA GeForce GTX 1650 Ti. The weights for the different CNN architectures were self-trained and saved in PTH format when used for DRF detection and DRF classification.

The batch sizes used varied and were tested multiple times to avoid any memory issues, and the epochs were set from a minimum of 35 to a maximum of 110 epoch. Adam algorithm was used after testing other optimizers and their effect on the performance, and the learning rate was set to 0.003 and 0.01 in multiple implementations. After training for the set number of epochs, the model validates and then saves the model.

The libraries used as mentioned before were Keras, PyTorch and TensorFlow with a varying version of python from 3.6 to the latest version to solve compatibility issues.

3.5 Results

This part is considered the most important throughout this thesis as it is the main summary of the work done in the previous semester.

3.5.1 Orientation Classification

As previously mentioned, this part used three main different models and different distribution of the dataset. The performance of the ResNet-50 and the VGG-16 proved to be dependent on the number of epochs and the data distribution given, especially when it came to the AP/Lat classification. The main goal of this section was to implement the highest performing model in both classifications with a strongly reliable accuracy, which in our case wound up to be the self-built CNN.

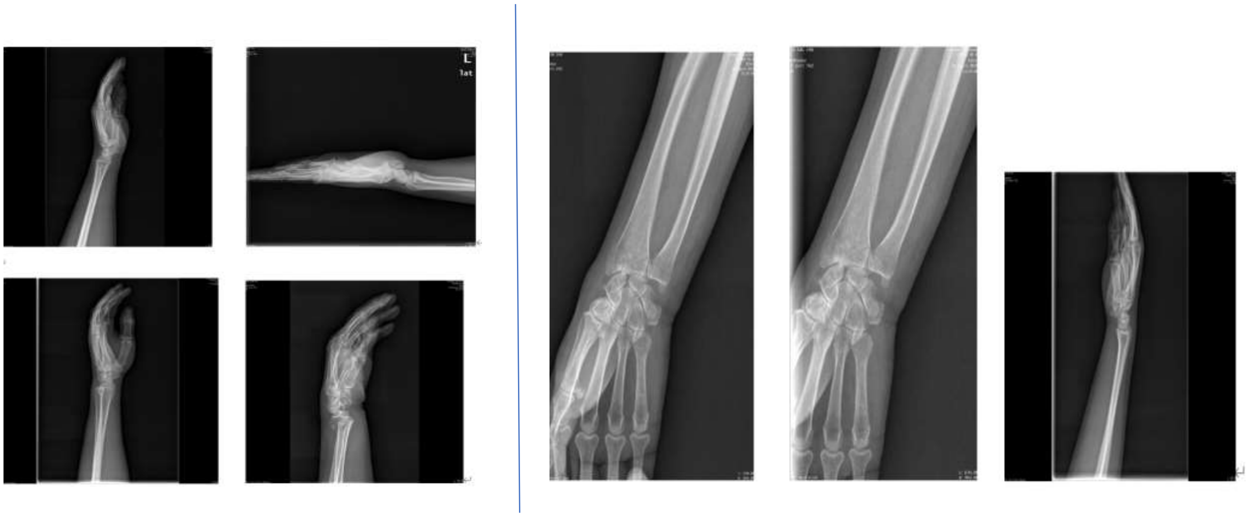
The following table in Fig. 3-12 summarizes the achieved top-1 accuracy in the 3 CNN models implemented during the orientation classification phase, regardless of number of epochs involved.

	AP/Lat Classification	Right/Left Classification	LR_AP Classification	LR_Lat Classification
ResNet-50	0.98	0.92	0.89	0.88
VGG-16	0.99	0.71	0.70	0.71
Our CNN	0.97	0.97	-	-

Fig. 3-12 comparative table of the top-1 accuracy in the employed CNN models.

As shown in Fig.3-12, we can see that the self-built CNN comes in the top place as the most reliable model in the orientation classification as a whole, run on a training session of 30 epochs alone. The ResNet-50 comes in second with a slightly lower accuracy when it comes to the Right/Left classification, while keeping in mind that the 92% accuracy was only achieved after running the model on 110 epochs, which is definitely time consuming. The ResNet-50 only reached an 84% accuracy when run on 30 epochs, thing that made it necessary to tweak the network for better performance. The VGG-16 came in last especially in the Left/Right classification, only reaching a 71% which made it rank last for the overall performance.

The table also shows the two columns where the dataset distribution was different. This approach was taken prior to the implementation of the self-built CNN, and after noticing the low accuracy on both ResNet-50 and VGG-16 when it came to the Left/Right hand classification, changing the dataset to a more precise one. This distribution was split to four sections where there is the Left-hand AP, Right-hand AP, Left-hand Lat, and Right-hand Lat. The concept seemed to be more precise and would make it easier for the networks to learn, but the performance shown was not reliable, even after taking different measures and model tweaking (data augmentation, optimizer change, loss function change, dropout tweak, number of epochs).



Correctly Classified Images **Incorrectly Classified Images**

Fig. 3-13 A comparative table of the top-1 accuracy in the employed CNN models.

It is necessary to add that the margin of error left was caused by incomplete hand images and some lateral ones. Fig. 3-13 shows examples of the classification results where the CNNs seemed to misclassify uncompleted images and one lateral one.

3.5.2 DRF Detection

The detection of the DRF is mostly important because it is the main focus of the project since the doctors need in their auxiliary diagnosis system a reliable detection system that manages to identify all the present distal radius fractures in one input radiograph. That is why three major widely used and popular CNN architectures were used and their performance was gauged accordingly to provide the most efficient detection.

Faster R-CNN, YOLO-v3 and YOLO-v4 were trained using the same dataset in PASCAL VOC07 with ROI training annotations in XML formats. Self-trained weights were used by running train scripts, with a torch version of 1.2.0. The performance of the models showed a comparable difference that took, not only the top-1 accuracy into account, but also memory usage and time consumed when run on a fixed number of epochs at 100.

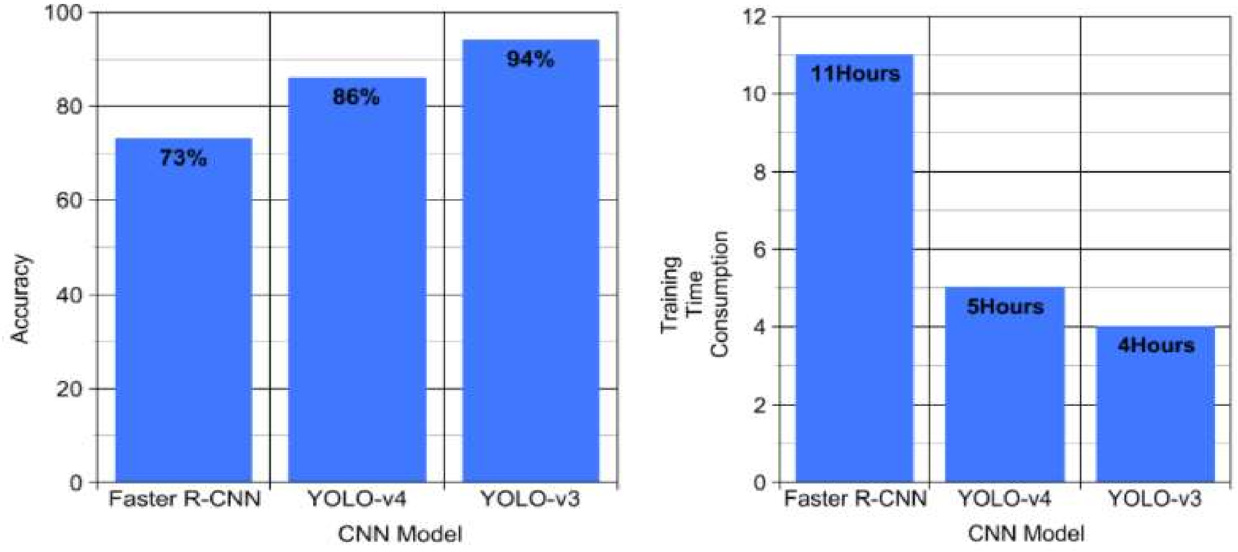


Fig. 3-14 Comparative graphs of the performance of the CNNs top-1 accuracy and time consumption wise.

As Fig. 3-14 shows, the best performance was showed by the YOLO-v3 model who not only achieved a top-1 accuracy of 94%, which is considered very high and reliable for our project, but also consumed the least time training the weights used, by only spending 1 to 3 minutes per epoch. YOLO-v4 showed a relatively high accuracy as well that can be reliable by reaching 86% and consuming only 4 hours with a 2 to 4 minutes per epoch. Lastly comes the performance of the Faster R-CNN which has only reached a 74% top-1 accuracy, and spending around 11 hours to train the 100 epochs, spending 6 to 9 min per each.

Memory wise, during the training, OOM errors were prompted multiple times, sometimes after spending long hours which made it necessary to reduce the batch size to a number compatible with the GPU’s memory.

An approach was used in this part of the project which is the freezing of the training. In other words, 50 epochs were available to train at first while the other 50 is frozen which created an initial training generation and a freezing training generation. This approach allows a speed up in the training speed and can also prevent the weight from being destroyed at the beginning of the training.

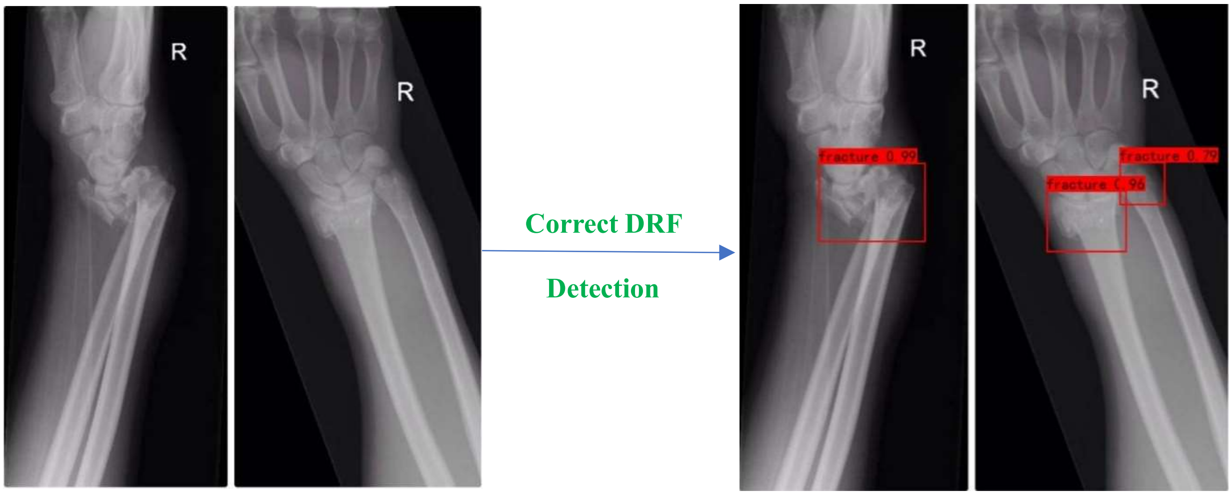


Fig. 3-15 Correct DRF Detection Results

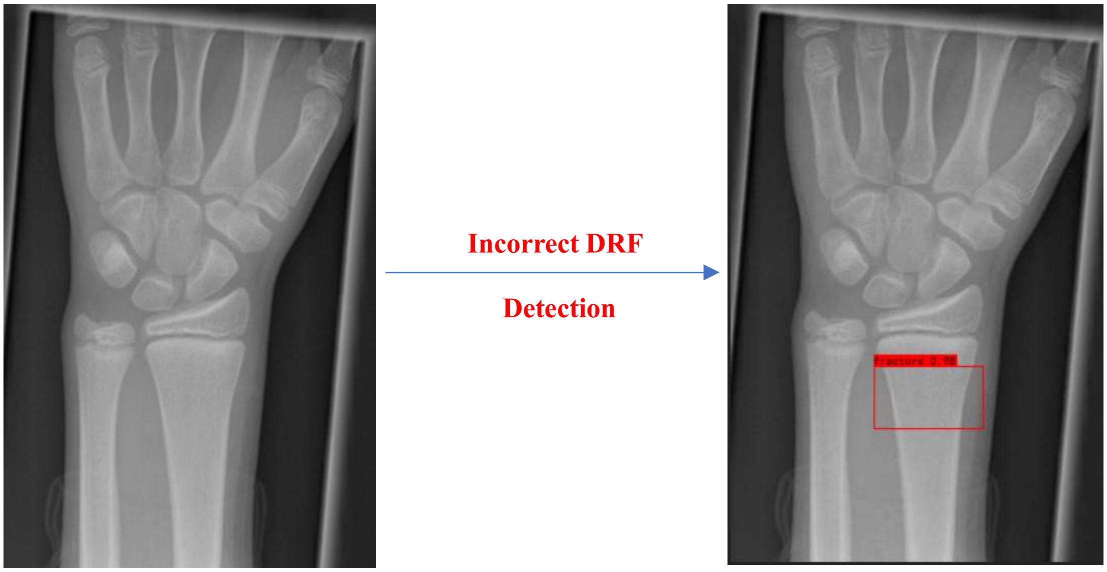
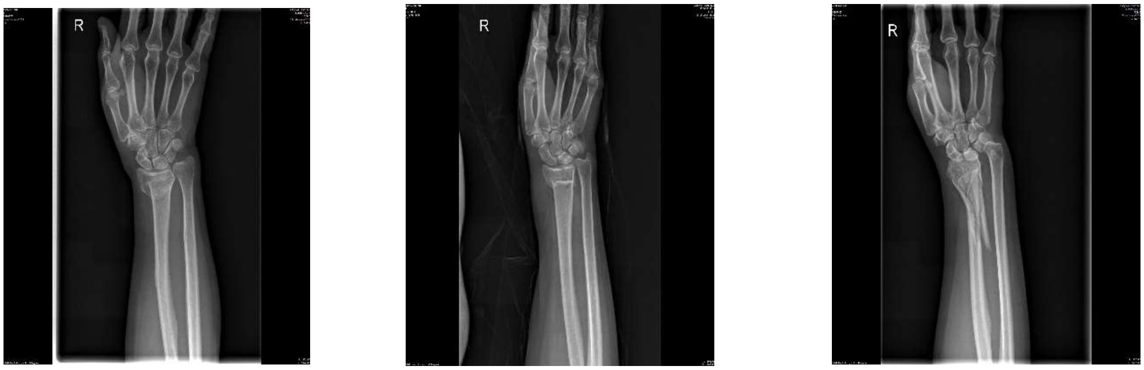


Fig. 3-16 Example Correct and Incorrect Detection Results, the incorrectness comes from the quality of the input radiograph.

3.5.3 DRF Classification



DRF Type A **DRF Type B** **DRF Type C**

Fig. 3-17 DRF Classification Sample Types

In the last part of our deep learning process, which takes it upon itself to categorize the types available of the DRF into Type A, B, and C, is using 4 CNN models. Two approaches were used during this part.

The first approach was basically not only a "Type" classification but a Type object detection. In other words, the model was fed a VOC07 with XML annotations and located ROIs then drew bounding boxes around the DRF while outputting its type instead of just detecting it as a fracture. A sample of the results can be shown below:

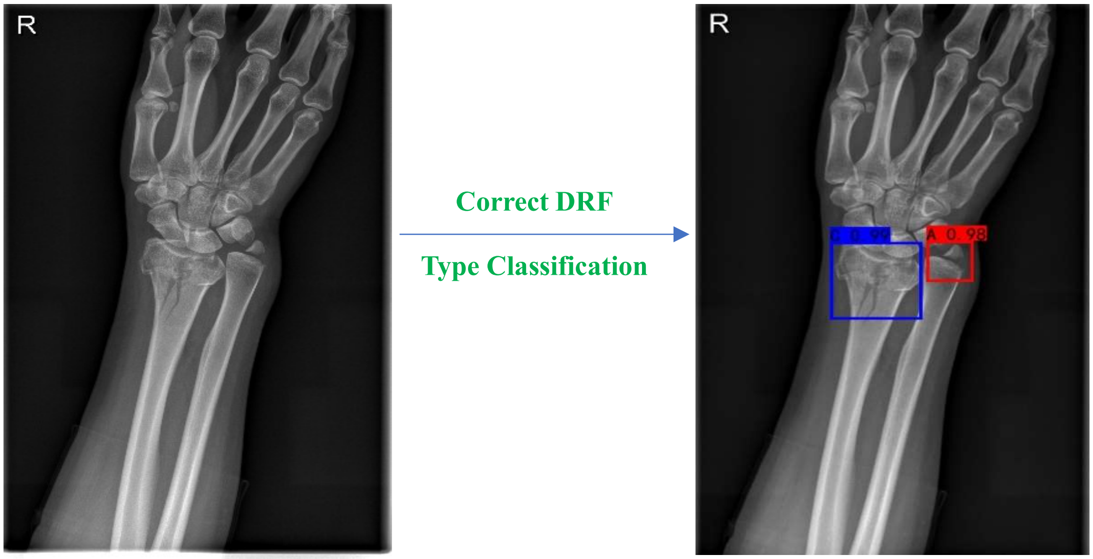


Fig. 3-18 Correct DRF Classification Results

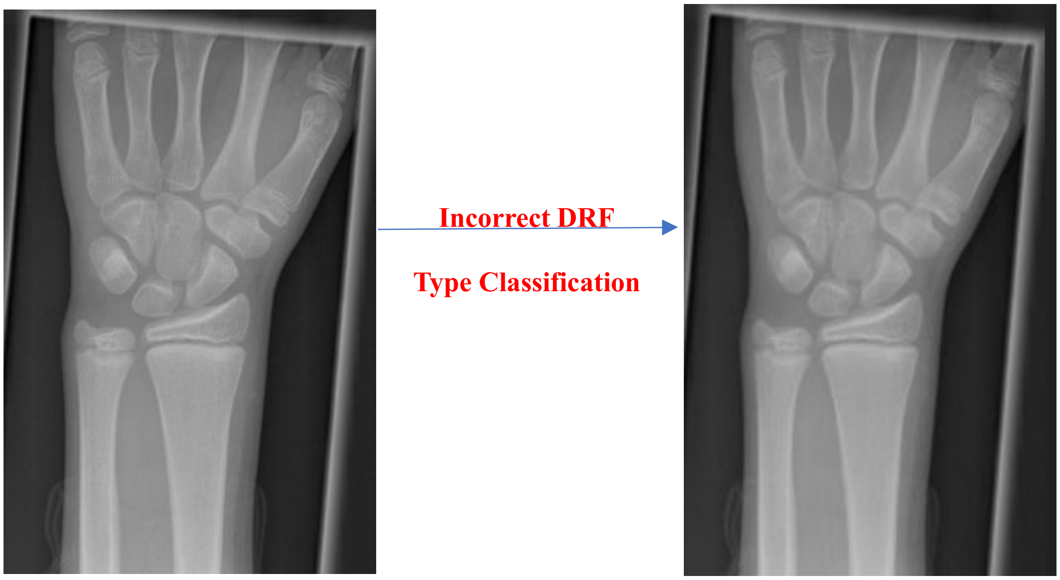


Fig. 3-19 Incorrect DRF Classification Results (the incorrectness comes from the quality of the input radiograph).

As the pictures show, the initial radiograph taken was output with detected fractures that were surrounded with bounding boxes and with the type selected instead, showing that the hand is suffering from both a Type A and a Type C DRFs. The same commentary can be said as mentioned in the object detection memory wise, similar number of epochs and same time consumption by the models.

The second approach was to take the dataset and repartition it into the classes we want (Type A, B, C) and run an image classification like the one performed during the orientation phase. We used the self-built CNN that showed astounding results in the first part. Data augmentation was performed on the dataset, zoom, vertical flip and rotation wise, and the time consumption was merely 10 minutes.

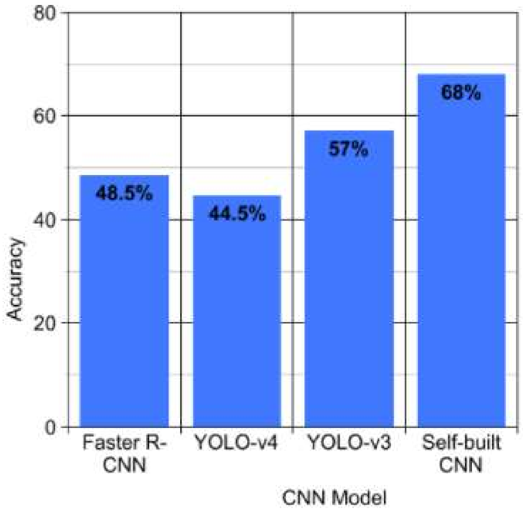


Fig. 3-20 Comparative graphs of the performance of the CNNs top-1 accuracy.

As Fig. 3-19 shows, the highest performance was achieved by the self-built CNN, even if it is not considered optimal, but still acceptable. DRF type detection wise, YOLO-v3 again showed to outperform the two other models with a 57% accuracy. The results were not high as expected which makes us question the reason for the confusion.

After further analysis, the conclusion was drawn that the reason behind the lack of efficient performance came from the inequality in the samples present in the graph as shown in the Dataset section. Type A almost the quadruple of the Type C and 7 times bigger than the Type B images available. The models did not have enough data to learn very well, but the results were still relatively reliable.

3.6 Dataset Analysis

As mentioned above, the models' performance was at times affected by the limitation in size as it did not always provide a large variety in type and orientation of the radiographs and DRF cases, which is certainly an important hurdle in the reliability of the system designed and results reached. Not to forget the huge difference found for example in the Type A, Type B, and Type C, datasets number wise as shown in Fig. 3-10 which limited the performance of the models when detecting those types of fractures.

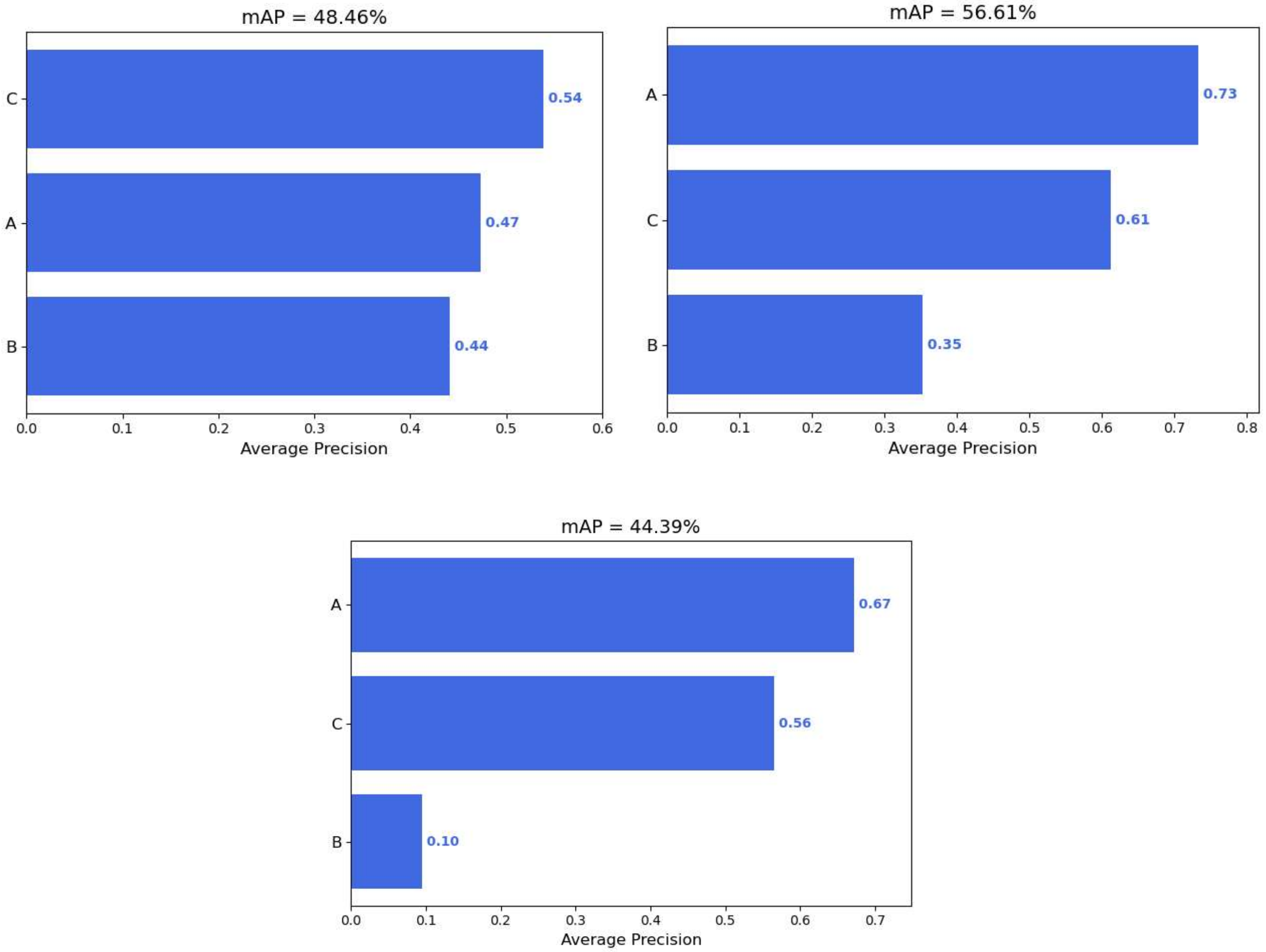


Fig. 3-21 Comparative graphs of the performance of the CNNs mAP for each DRF Type.

As shown by the mean average precision (mAP) graphs above of the three models, Type B always ranked last and this is clearly due to the lack of recourses given in its type specifically, while Type A was always the one with the highest recognition as it had more than 800 radiographs involved, compared to the 268 images of Type C and 105 images of Type B.

The bigger and wider the datasets used in this project, the better the performance achieved, and this only calls for better cooperation with hospitals and specialists to provide more reliable big data in the healthcare system and in our case, DRFs.

Chapter 4. System Design and Implementation

4.1 Overview

The second main part of this thesis, after the deep learning part, is the design and implementation of a mobile application that acts as a DRF diagnosis system that can be available in worldwide hospitals. Our mobile DRF diagnose will provide a fast tool for doctors and medics, to snap a picture of the radiograph and get instant diagnostic results. This of course serves its main purpose of saving doctors from repetitive boring work, and fasten the procedure of the diagnosis, especially in ERs where DRFs are very common cases.

4.2 Basic Information and System Implementation

4.2.1 Language

The system was developed by Flutter and Dart. Flutter is a Google-created free and open-source mobile UI framework that was introduced in May 2017. In a nutshell, it enables you to develop a native mobile app using only one codebase. This implies you can design two different apps using the same programming language and codebase (for iOS and Android).

Flutter is made up of two main components:

- An SDK (Software Development Kit) is a set of tools that will assist you in the development of your applications. Tools for compiling your code into native machine code are included (code for iOS and Android).
- A Framework (Widget-based UI Library): A set of reusable user interface elements (buttons, text inputs, sliders) that you may customize to fit your needs.

The Dart programming language is utilized to create our Flutter app. Google established the language in October 2011, but it has come a long way in the last several years. Dart is a front-end programming language that may be used to create mobile and online applications.

The main reason behind choosing Flutter to implement our DRF Diagnosis System is the fact that it will allow the app to run on both iOS and Android phones, which will save time and make it available for almost all types of smart phones.

4.2.2 Main Concept

The way the application works is that it allows the user to enter the patient's information, select a picture either from his/her phone Gallery or take an instant photo of the radiograph. After confirmation, the background process is triggered which consists of our Deep Learning model deployed to classify the given radiograph orientation wise, detect the DRF and classify it into its correct type. It then returns a diagnosis page filled with the needed information, allowing the user to take the treatment decision right away. A database implementation is included and keeps track of the patients' medical records for easy access by the responsible party. Fig. 4-1 summarizes the overall functions of the mobile DRF diagnosis system as shown below.

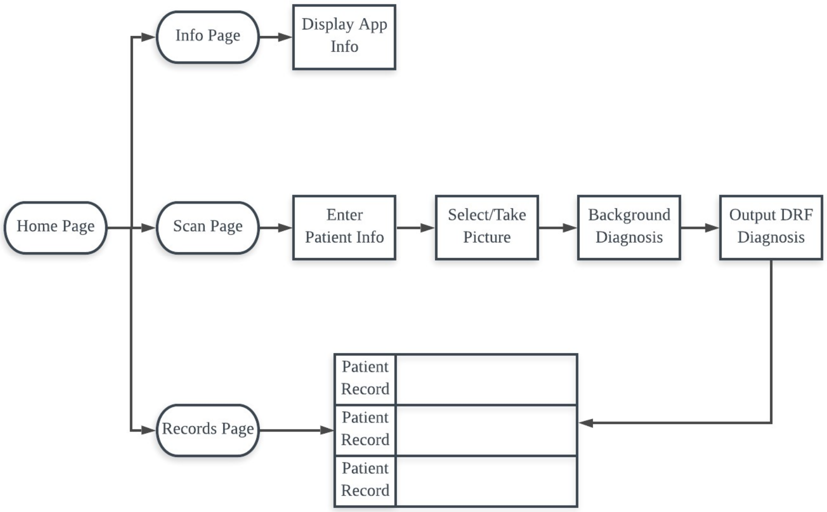
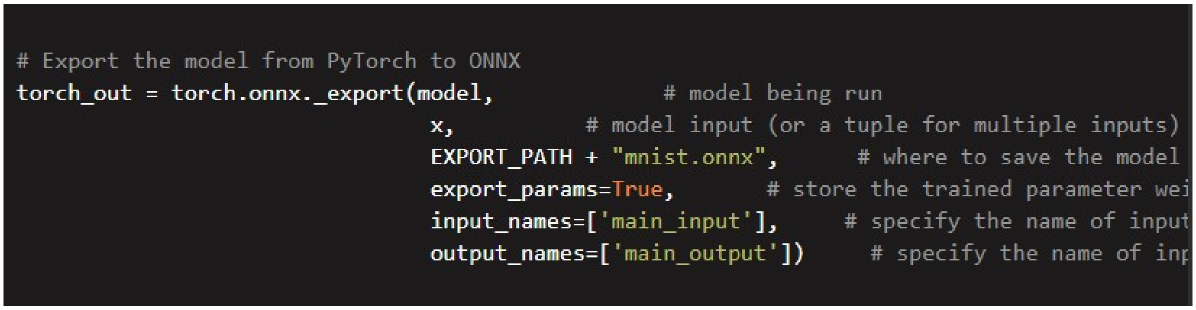


Fig. 4-1 Global DRF Diagnosis Mobile App Flowchart

4.2.3 Models Deployment

Our models are implemented in multiple deep learning libraries as mentioned before, which makes the compatibility between our models and our app to pose an important hurdle. Our models in all phases needed to be converted to the .tflite format. The conversion for Keras saved models in .h5 format was easy and direct by using built-in methods. The conversion from PyTorch to TensorflowLite takes extra steps which can be summed in the following steps:

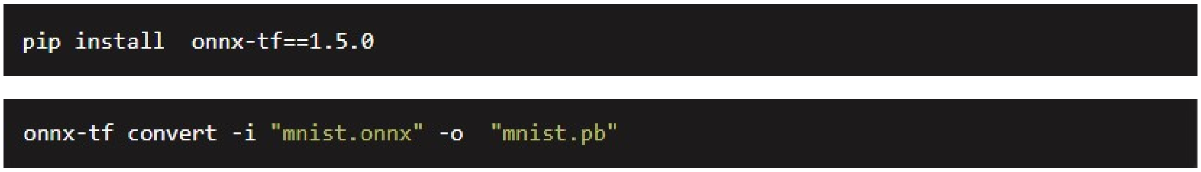
- Convert from PyTorch to ONNX model



```
# Export the model from PyTorch to ONNX
torch_out = torch.onnx._export(model,             # model being run
                                x,          # model input (or a tuple for multiple inputs)
                                EXPORT_PATH + "mnist.onnx",      # where to save the model
                                export_params=True,       # store the trained parameter wei
                                input_names=['main_input'],     # specify the name of input
                                output_names=['main_output'])     # specify the name of inp
```

Fig. 4-2 Python Code for PyTorch-ONNX Conversion.

- Convert from ONNX to TensorFlow FreezeGraph

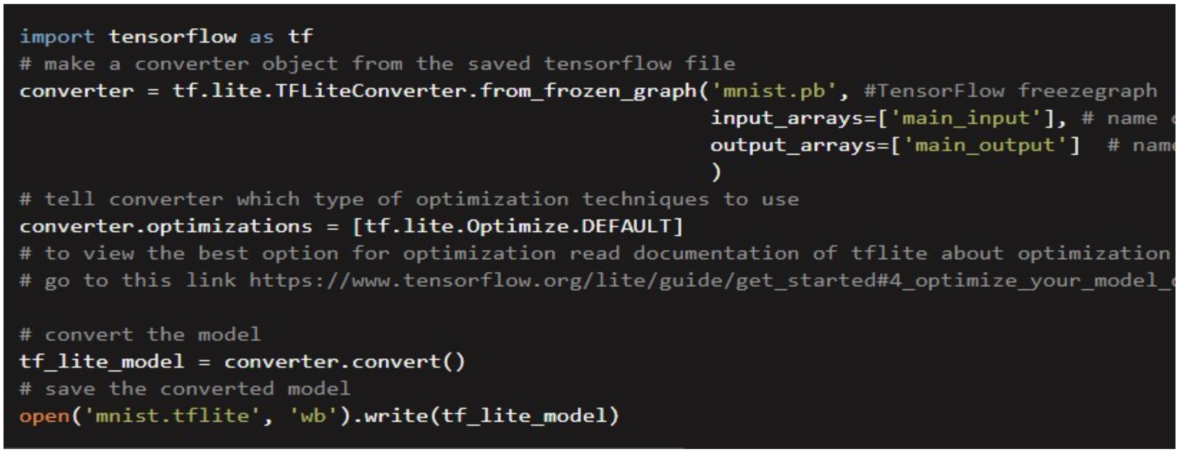


```
pip install  onnx-tf==1.5.0
```

```
onnx-tf convert -i "mnist.onnx" -o  "mnist.pb"
```

Fig. 4-3 Python Code for ONNX-TF FreezeGraph Conversion.

- Convert from TensorFlow FreezeGraph .pb to .tflite



```
import tensorflow as tf
# make a converter object from the saved tensorflow file
converter = tf.lite.TFLiteConverter.from_frozen_graph('mnist.pb', #TensorFlow freezegraph
                                                      input_arrays=['main_input'], # name
                                                      output_arrays=['main_output']  # nam
                                                      )
# tell converter which type of optimization techniques to use
converter.optimizations = [tf.lite.Optimize.DEFAULT]
# to view the best option for optimization read documentation of tflite about optimization
# go to this link https://www.tensorflow.org/lite/guide/get_started#4_optimize_your_model_

# convert the model
tf_lite_model = converter.convert()
# save the converted model
open('mnist.tflite', 'wb').write(tf_lite_model)
```

Fig. 4-4 Python Code for FreezeGraph-TFLite Conversion.

This guide proved to be the best way found to convert PyTorch models to TensorflowLite and deploy them accordingly in the Flutter app without issues. **Teachable machine** also provides a fast alternative, by creating your classes, adding the images, and training the model with customized epochs and learning rates, testing it and then saving it straight to .tflite format, but this can only be used in image classification.

The limitations presented when deploying our CNN models is that their conversion affects the overall accuracy and reliability of the model, especially when it came to object detection. Multiple tensor compatibility errors were generated over the app implementation which took lots of time into debugging and trying to solve them. The fact that Flutter is relatively new poses a problem too as there are not enough tutorials or solutions online to help us through the process successfully. This calls for implementing a specialized way to do the conversion successfully without getting compatibility errors and have the accuracy and performance of our models stay the same. Solving this main issue is crucial as it will allow us to link the DRF detection directly into the app like the classification models, without having the import the results from the python files’ execution.

4.2.4 Database

Since keeping a record of the medical information of the patients already diagnosed by the app, setting up a database and linking it was very important. Our database was implemented based on SQLite. SQLite is a relational database management system implemented as a software library. In terms of setup, database administration, and required resources, the lite in SQLite stands for "lightweight." The **sqflite** plugin, which is accessible on **pub.dev**, allows Flutter apps to access SQLite databases. This recipe uses the following steps:

1. Add the dependencies.
2. Define the Record data model.
3. Open the database.
4. Create the Record table.
5. Insert a Record into the database.
6. Retrieve the list of Records.
7. Delete a Record from the database.

Our database elements consist of the patient’s name, the patient`s medical ID, the patient’s Sex, and the patient’s diagnosis summary. The first three elements were used successfully to store the medical records, but the diagnosis summary paused a privacy issue. In other words, since the system already takes in the medical ID that the responsible party could use to look up the patient’s medical history, and access the diagnosis via trusted devices, we found it unnecessary to store the diagnosis while not having a strong security and privacy base in our Mobile Application, which is not trust-worthy.

This database element can only be used after figuring out strong privacy base to protect the patients all around the world. Additional elements were included but they mainly acted as features for the UI and usability side. Table 4-1 Shows the Record Table Structure.

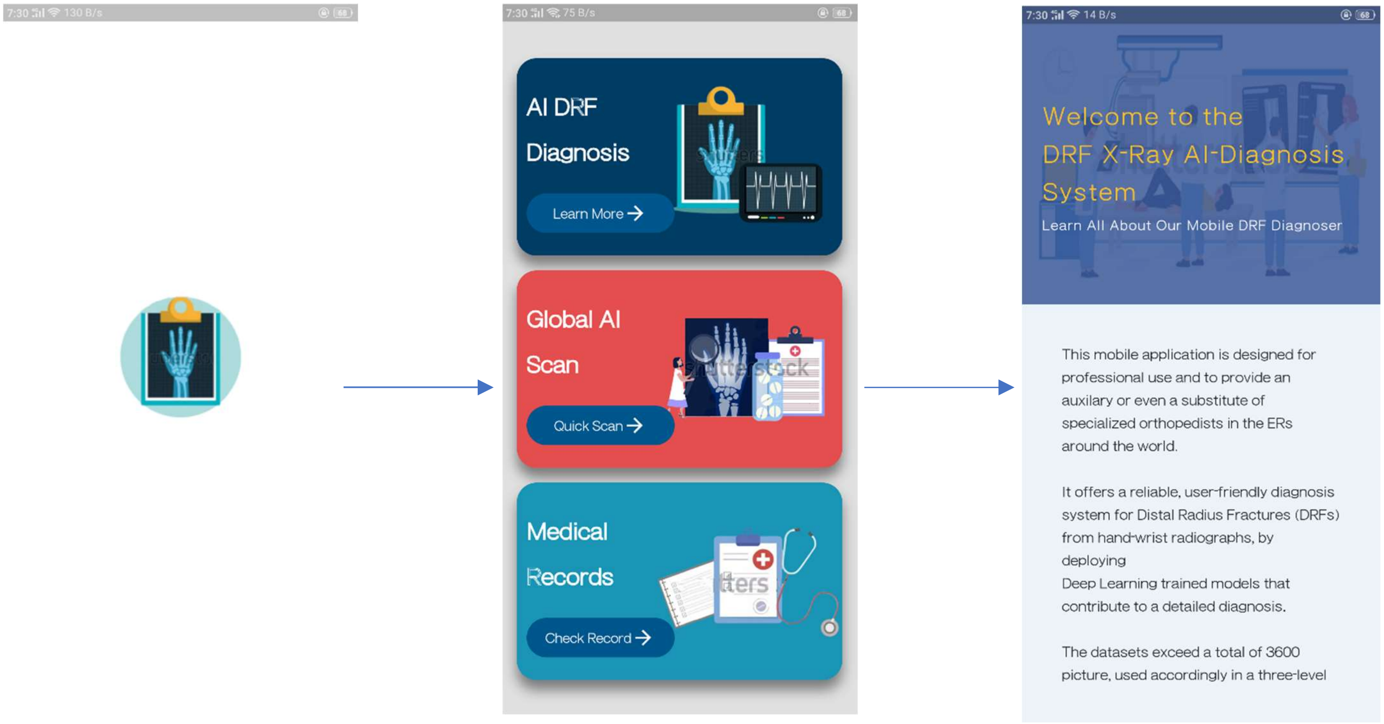
Table 4-1 Medical Record database Table Elements

Field	Data Type	Key
id	Integer auto-increment	Primary key
patientName	Text not null	-
patientId	Text not null	-
recordDateTime	Text not null	-
Sex	Text not null	-
gradientColorIndex	Integer	-
Diagnosis	Text not null	-

4.3 DRF Diagnosis App Interface

The following part focuses on displaying the user interface of the discussed activities shown in Fig. 4-5 and discuss the main features they have. Our DRF Diagnosis mobile app is named **Diagno**. When first entering the app, the splash screen is displayed first and then the user is led to the home page that hosts the main activities. The user has three main options.

The first one being clicking the button Learn More in AI DRF Diagnosis in order to navigate to the info page that explains the main points concerning the mobile App. It is used to include the user into the perspective of how the app was developed, some info about the datasets and an overall brief discussion on how it works.



Welcome to the DRF X-Ray AI-Diagnosis System

Learn All About Our Mobile DRF Diagnoser

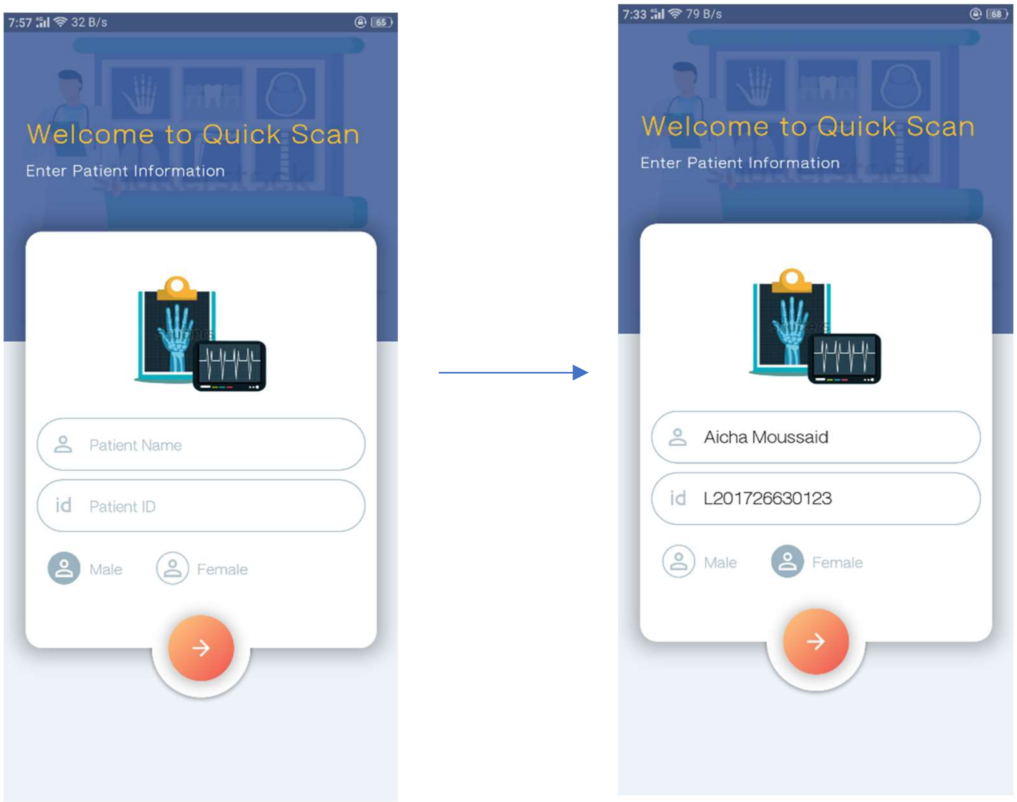
This mobile application is designed for professional use and to provide an auxilary or even a substitute of specialized orthopedists in the ERs around the world.

It offers a reliable, user-friendly diagnosis system for Distal Radius Fractures (DRFs) from hand-wrist radiographs, by deploying Deep Learning trained models that contribute to a detailed diagnosis.

The datasets exceed a total of 3600 picture, used accordingly in a three-level

Fig. 4-5 Diagno Splash, Home and Info Interfaces.

The second option is to click on the Quick Scan button to navigate to the Global AI Scan, and this is where our app mainly fulfills its diagnostic purpose. The user is prompted to enter the patient’s information, is led to the page where he can pick or take a picture, followed by a loading page that takes him/her to the diagnosis results page.

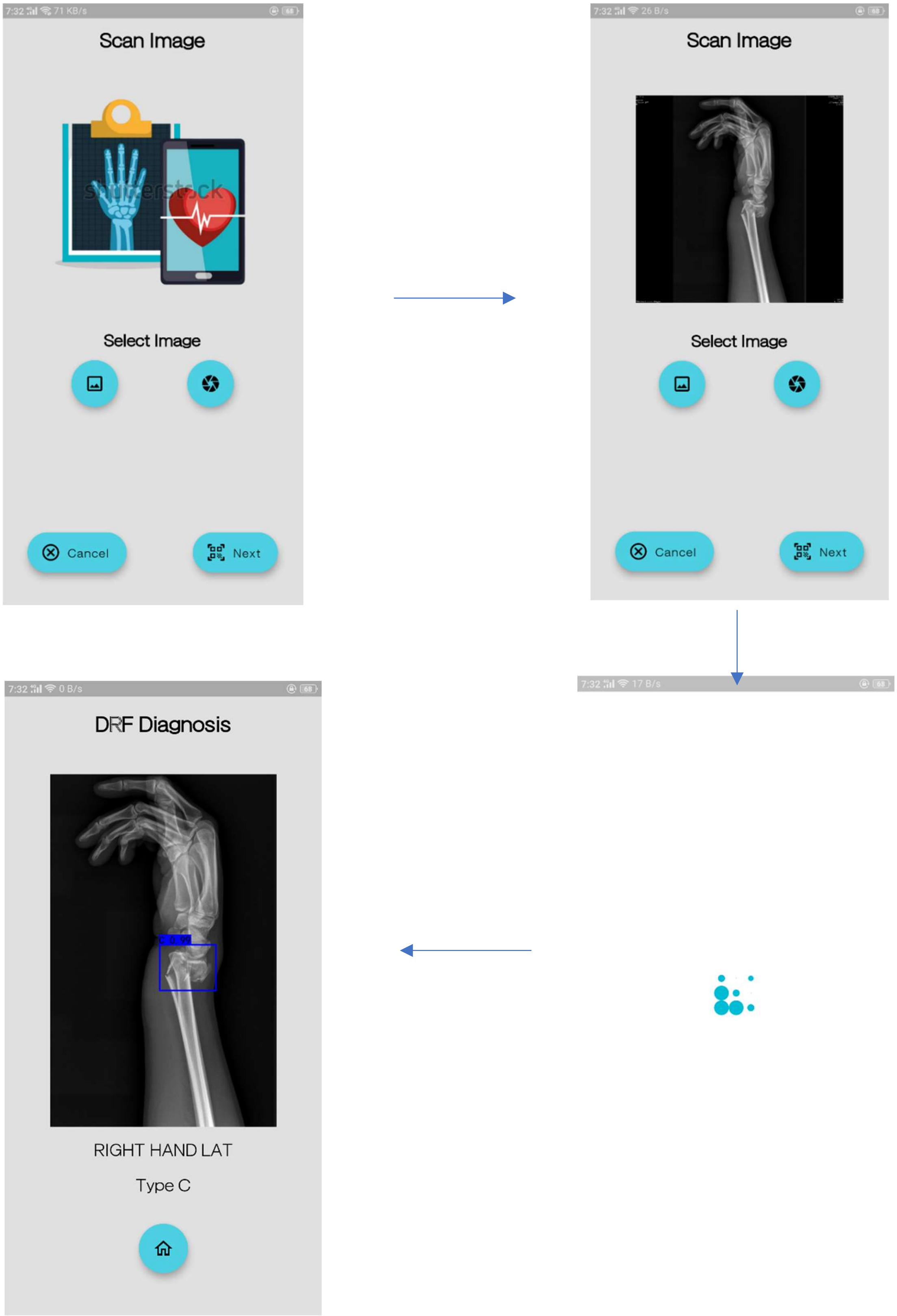


Aicha Moussaid

id L201726630123

Fig. 4-6 Diagno Patient Info Interface

The user input validation has an awaiting feature that will be linked to the nature of the hospital. In other words, each hospital has a specific medical ID reference given to its users, depending on the country, the hospital, etc. This leads us to implement the ID validation compatible with the medics using it. Also, the input validation, after figuring out a strong privacy and security system of the app, shall confirm that there is an existing patient with that ID and name in the system, so that the storage of the diagnosis will be direct.



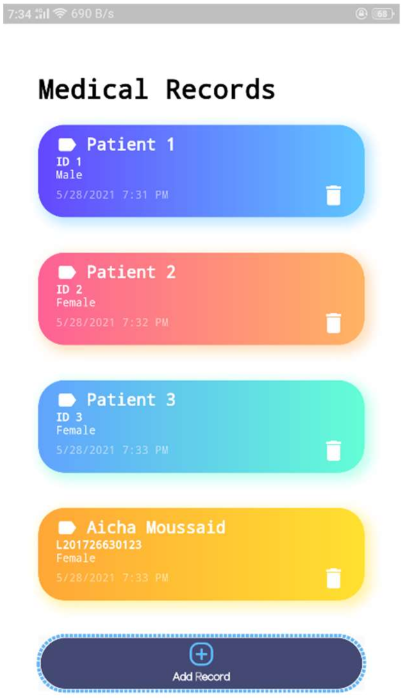
RIGHT HAND LAT

Type C

Fig. 4-7 Diagno Scan Interface.

The user has two options when first led to the Scan page, either choose the picture from the gallery or take an instant shot. After confirmation of the selection, the page is updated to holding the image selected and can now click on Next button for the diagnosis to be performed. After seconds of loading, the user gets to see the final results, which hold an updated image with the fracture detected and the ROI highlighted with bounding boxes, the results of both the orientation and the DRF type classifications. The user can then click the Home button to navigate back to the Home Screen.

After this whole procedure, the user can find the records in the Medical Records page by clicking on Check Records button. As shown in the screenshot bellow, the records are displayed, with the correct information held in them. The user is given the ability to either delete the records permanently from the database or Add a new record which leads straight to the Scan page.



Medical Records

Patient 1
ID 1
Male
5/28/2021 7:31 PM

Patient 2
ID 2
Female
5/28/2021 7:32 PM

Patient 3
ID 3
Female
5/28/2021 7:33 PM

Aicha Moussaid
L201726630123
Female
5/28/2021 7:33 PM

Fig. 4-8 Diagno Medical Records Interface.

4.4 Scope of Improvement

This project is far from perfection and holds a large scope of improvement. If we ignore the accuracy increase possible with better datasets and better model conversion, and focus on the mobile application, the scope of development and enrichment is certainly important. This app can turn from a one specialty diagnostic system to a radiology global one.

In other words, given a strong contribution and a wider team, both specialized in Artificial Intelligence and Mobile App Development, this app can cover a wide range of diagnosis systems and automatize them in anything radiology involved. Not only this but the possible platforms it can be implemented in can revolutionize the daily tasks of doctors and medics and improve the healthcare system as a whole.

Not to forget the necessity for a strong reliable privacy base that protects both our users and our patients' information and linking it to the hospital's own database so that the results can be checked on any other platform available for the medics and doctors concerned. This can make it easier for the treatment specialized unit to straight check the results of the diagnosis and straight lead the needed procedure.

Chapter 5. Conclusion

5.1 Summary

In this thesis, a global study was led to compare and look for the optimal implementation of a reliable Distal Radius Fracture Diagnosis System based on deep learning.

We implemented a three-level process deep learning pipeline which performs an orientation classification of the radiographs, a DRF detection and a DRF classification, using diverse Convolutional Neural Network architectures and analyzing their performances.

We then proceeded to create a system which deploys the models saved resulting from the previous part and implemented a mobile application that can be easily used and accessible by doctors and medics in the ER.

As evidenced by the discussion of the results obtained, this project has a huge scope of improvement that can be achieved with more sophisticated designs of networks and given larger training datasets with equal amounts or sufficiently equal ranges in all classes, which can push the models to reach 100% accuracy and the mobile app to be a solid diagnostic system available in all hospitals.

5.2 Future Works

As afore mentioned in the scope of improvement in Chapter 4, this project has a large margin of advancement. Our future works include a new implementation for DRF Type classification by designing our own object detection CNN that can achieve higher accuracies, while providing a wider, balanced range of data for training. The performance hoped for should surpass that of radiologists and orthopedists, achieving a 100% accuracy, because the medical field does not leave margin for error as it is critical for patients.

Our plans also include improvements in Diagno, our mobile application, concerning the user-input validation, implement a server side to run the background deep learning process without conversion, and broaden its horizons to becoming a wider radiology related app, where multiple radiograph diagnosis can be performed. This of course requires a team-oriented approach with specialists and lots of testing, in order to provide the hospitals and the medics of a reliable trust-worthy diagnosis systems, devoid of errors.

References

[1] FRAMINGHAM, Mass., August 25, 2020, Worldwide Spending on Artificial Intelligence Is Expected to Double in Four Years, Reaching $110 Billion in 2024, According to New IDC Spending Guide

[2] "Distal radius fractures are difficult to classify." Wæver D, Madsen ML, Rölfing JHD, Borris LC, Henriksen M, Nagel LL, Thorninger R Injury. 2018 Jun; 49 Suppl 1():S29-S32.
[3] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. 2015 IEEE International Conference on Computer Vision; 2015.

[4] Russakovsky O, Deng J, Su H, krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A C, Li F F. ImageNet large scale visual recognition challenge. Int J Comput Vis 2015;

[5] Russakovsky O, Deng J, Su H, krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A C, Li F F. ImageNet large scale visual recognition challenge. Int J Comput Vis 2015; 115(3): 211–52.

[6] Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks.Lakhani P, Sundaram B Radiology. 2017 Aug; 284(2):574-582.

[7] Computer-aided classification of lung nodules on computed tomography images via deep learning technique. Hua KL, Hsu CH, Hidayati SC, Cheng WH, Chen YJ Onco Targets Ther. 2015; 8():2015-22.

[8] Automated detection and classification of the proximal humerus fracture by using deep learning algorithm. Chung SW, Han SS, Lee JW, Oh KS, Kim NR, Yoon JP, Kim JY, Moon SH, Kwon J, Lee HJ, Noh YM, Kim Y Acta Orthop. 2018 Aug; 89(4):468-473.

[9] Artificial intelligence in fracture detection: transfer learning from deep convolutional neural networks. Kim DH, MacKinnon T Clin Radiol. 2018 May; 73(5):439-445.

[10] Detecting intertrochanteric hip fractures with orthopedist-level accuracy using a deep convolutional neural network. Urakawa T, Tanaka Y, Goto S, Matsuzawa H, Watanabe K, Endo N Skeletal Radiol. 2019 Feb; 48(2):239-244.

[11] Automated detection and classification of the proximal humerus fracture by using deep learning algorithm. Chung SW, Han SS, Lee JW, Oh KS, Kim NR, Yoon JP, Kim JY, Moon SH, Kwon J, Lee HJ, Noh YM, Kim Y Acta Orthop. 2018 Aug; 89(4):468-473.

[12] Detecting intertrochanteric hip fractures with orthopedist-level accuracy using a deep convolutional neural network. Urakawa T, Tanaka Y

[13] Suzuki, K. Overview of deep learning in medical imaging. *Radiol Phys Technol* **10,** 257–273 (2017). https://doi.org/10.1007/s12194-017-0406-5

[14] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, Namkug Kim, Korean J Radiol. 2017 Jul-Aug; 18(4): 570–584. Published online 2017 May 19. doi: 10.3348/kjr.2017.18.4.570 PMCID: PMC5447633

[15] Asifullah Khan, , Anabia Sohail, Umme Zahoora and Aqsa Saeed Qureshi. A Survey of the Recent Architectures of Deep Convolutional Neural Networks

[16] Fakultät für , und Naturwissenschaften Lehr- und Forschungsgebiet "Understanding Convolutional Neural Networks", VIII Computer Vision Prof.

[17] Artificial intelligence in fracture detection: transfer learning from deep convolutional neural networks. Kim DH, MacKinnon T Clin Radiol. 2018 May; 73(5):439-445.

[18] Artificial intelligence for analyzing orthopedic trauma radiographs. Olczak J, Fahlberg N, Maki A, Razavian AS, Jilert A, Stark A, Sköldenberg O, Gordon M. Acta Orthop. 2017 Dec; 88(6):581-586.

[19] Automated detection and classification of the proximal humerus fracture by using deep learning algorithm. Chung SW, Han SS, Lee JW, Oh KS, Kim NR, Yoon JP, Kim JY, Moon SH, Kwon J, Lee HJ, Noh YM, Kim Y Acta Orthop. 2018 Aug; 89(4):468-473.

[20] Detecting intertrochanteric hip fractures with orthopedist-level accuracy using a deep convolutional neural network. Urakawa T, Tanaka Y, Goto S, Matsuzawa H, Watanabe K, Endo N.Skeletal Radiol. 2019 Feb; 48(2):239-244.

[21] "Artificial intelligence detection of distal radius fractures: a comparison between the convolutional neural network and professional assessments" Kaifeng Gan, Dingli Xu, Yimu Lin, Yandong Shen, Ting Zhang, Keqi Hu, Ke Zhou, Mingguang Bi, Lingxiao Pan, Wei Wu, and Yunpeng Liu, Acta Orthop. 2019 Aug; 90(4): 394–400. Published online 2019 Apr 3. doi: 10.1080/17453674.2019.1600125,

[22] "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. arXiv:1506.01497

[23] "YOLOv3: An Incremental Improvement" Joseph Redmon, Ali Farhadi University of Washington

[24] "YOLOv4: Optimal Speed and Accuracy of Object Detection" Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao arXiv:2004.10934v1 [cs.CV] 23 Apr 2020

[25] “Distal radius fractures: Still a common problem” C. Acosta-OlivoTraumatology and Orthopedics, “Dr. José E. González” University Hospital, Autonomous University of Nuevo Leon, Monterrey, Mexico Vol. 19. Issue 76. pages 140-142 (July - September 2017)

[26] VOLUME 126, 108925, MAY 01, 2020 Detection and localization of distal radius fractures: Deep learning system versus radiologists, Christian Blüthgen, Anton S. Becker, Ilaria Vittoria de Martini, Andreas Meier, Katharina Martini, Thomas Frauenfelder

[27] Sander A.L. Leiblein M. Sommer K. Marzi I. Schneidmuller D. Frank J. “Epidemiology and treatment of distal radius fractures: current concept based on fracture severity and not on age”. Eur. J. Trauma Emerg. Surg. 2018;

[28] Karen Simonyan, Andrew Zisserman “Very Deep Convolutional Networks for Large-Scale Image Recognition” arXiv:1409.1556

[29] Shilpa Ananth, “Faster R-CNN for object detection, A technical paper summary”, Aug 9, 2019, < https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46 >

Acknowledgement

First of all, I would like to thank my family for the endless support, both financial and emotional, and for encouraging me and pushing me to be better professionally and academically. My undergraduate studies would not have been possible, especially in China, if it were not for their sacrifice.

Equally, I would like to thank Hao Pengyi for being an amazing teacher that taught me valuable lessons and for being my supervisor. Her trust in me and her support provided whenever I needed clarity made her a strong base to the completion of this thesis successfully. She is someone I will always be grateful for, and make sure to always look up to as she is a perfect example of an academically successful woman.

I also would like to thank both my friends in Morocco, and the friends I made during my stay in China for their constant support, and for making my life that much better and balancing it out.

Not to forget a special thank you note to myself for making it all the way, stepping over all the hurdles and making it work, no matter circumstances.