ΠΑΝΕΠΙΣΤΗΜΙΟΔΥΤΙΚΗΣΑΤΤΙΚΗΣUNIVERSITYOFWESTATTICA
ΣΧΟΛΗΜΗΧΑΝΙΚΩΝFACULTYOFENGINEERING
ΤΜΗΜΑΗΛΕΚΤΡΟΛΟΓΩΝΚΑΙDEPARTMENTOFELECTRICAL&
ΗΛΕΚΤΡΟΝΙΚΩΝΜΗΧΑΝΙΚΩΝELECTRONICSENGINEERING

ΣΧΕΔΙΑΣΗΣΚΑΙΠΑΡΑΓΩΓΗΣ
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr
Θηβών 250, Αθήνα-Αιγάλεω 12241
Τηλ: +30 210 538-1614

PRODUCTIONENGINEERING
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr
250, Thivon Str., Athens, GR-12241, Greece
Tel: +30 210 538-1614

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών
Τεχνητή Νοημοσύνη και Βαθιά Μάθηση
https://aidl.uniwa.gr/

Master of Science in
Artificial Intelligence and Deep Learning
https://aidl.uniwa.gr/

ΤΜΗΜΑΜΗΧΑΝΙΚΩΝΒΙΟΜΗΧΑΝΙΚΗΣDEPARTMENTOFINDUSTRIALDESIGNAND

Master of Science Thesis

Artificial Intelligence in medical diagnosis (with emphasis on orthopedics)

Student: Skaleris, Stamatios-Michail
Registration Number: AIDL-0032

MSc Thesis Supervisor

Leligkou, Eleni Aikaterini
Professor

ATHENS-EGALEO, September 2024

ΣΧΕΔΙΑΣΗΣΚΑΙΠΑΡΑΓΩΓΗΣ
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr

PRODUCTIONENGINEERING
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr

Θηβών 250, Αθήνα-Αιγάλεω 12241
Τηλ: +30 210 538-1614
Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών
Τεχνητή Νοημοσύνη και Βαθιά Μάθηση
https://aidl.uniwa.gr/

250, Thivon Str., Athens, GR-12241, Greece
Tel: +30 210 538-1614
Master of Science in
Artificial Intelligence and Deep Learning
https://aidl.uniwa.gr/

ΠΑΝΕΠΙΣΤΗΜΙΟΔΥΤΙΚΗΣΑΤΤΙΚΗΣUNIVERSITYOFWESTATTICA
ΣΧΟΛΗΜΗΧΑΝΙΚΩΝFACULTYOFENGINEERING
ΤΜΗΜΑΗΛΕΚΤΡΟΛΟΓΩΝΚΑΙDEPARTMENTOFELECTRICAL&
ΗΛΕΚΤΡΟΝΙΚΩΝΜΗΧΑΝΙΚΩΝELECTRONICSENGINEERING
ΤΜΗΜΑΜΗΧΑΝΙΚΩΝΒΙΟΜΗΧΑΝΙΚΗΣDEPARTMENTOFINDUSTRIALDESIGNAND

Μεταπτυχιακή Διπλωματική Εργασία

Τεχνητή νοημοσύνη στην ιατρική διάγνωση (με έμφαση στην ορθοπεδική)

Φοιτητής: Σκαλέρης, Σταμάτιος-Μιχαήλ
ΑΜ: AIDL-0032

Επιβλέπουσα Καθηγήτρια
Λελίγκου, Ελένη Αικατερίνη
Καθηγήτρια

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, Σεπτέμβριος 2024

This MSc Thesis has been accepted, evaluated and graded by the following committee:

Supervisor	Member	Member
ELENI AIKATERINI LELIGKOU Digitally signed by ELENI AIKATERINI LELIGKOU Date: 2024.11.08 15:41:28 +02'00'	Digitally signed by CHARALAMPOS PATRIKAKIS Date: 2024.11.09 18:30:50 +02'00'	GEORGIOS GEORGOUDIS Digitally signed by GEORGIOS GEORGOUDIS Date: 2024.11.09 16:41:06 +02'00'
Eleni Aikaterini (Nelly) Leligou	Chalampos Patrikakis	George Georgoudis
Professor	Professor	Professor
Industrial Design and Production Engineering Department	Electrical And Electronic Engineering Department	Physiotherapy Department
University of West Attica	University of West Attica	University of West Attica

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Σταμάτιος-Μιχαήλ Σκαλέρης. του Ιωάννη, με αριθμό μητρώου mscaidl-0032 μεταπτυχιακός φοιτητής του ΔΠΜΣ «Τεχνητή Νοημοσύνη και Βαθιά Μάθηση» του Τμήματος Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών και του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής, της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, **δηλώνω υπεύθυνα ότι:**

«Είμαι συγγραφέας αυτής της μεταπτυχιακής διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Η εργασία δεν έχει κατατεθεί στο πλαίσιο των απαιτήσεων για τη λήψη άλλου τίτλου σπουδών ή επαγγελματικής πιστοποίησης πλην του παρόντος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …… και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος/ουσας καθηγητή/ήτριας.»

Ο/Η Δηλών/ούσα

Σταμάτιος-Μιχαήλ Σκαλέρης

Declaration of the author of this MSc thesis

I, Stamatios-Michail Skaleris of Ioannis with the following student registration number: mscaidl-0032, postgraduate student of the MSc programme in "Artificial Intelligence and Deep Learning", which is organized by the Department of Electrical and Electronic Engineering and the Department of Industrial Design and Production Engineering of the Faculty of Engineering of the University of West Attica, hereby declare that:

I am the author of this MSc thesis and any help I may have received is clearly mentioned in the thesis. Additionally, all the sources I have used (e.g., to extract data, ideas, words or phrases) are cited with full reference to the corresponding authors, the publishing house or the journal; this also applies to the Internet sources that I have used. I also confirm that I have personally written this thesis and the intellectual property rights belong to myself and to the University of West Attica. This work has not been submitted for any other degree or professional qualification except as specified in it.

Any violations of my academic responsibilities, as stated above, constitutes substantial reason for the cancellation of the conferred MSc degree.

I wish to deny access to the full text of my MSc thesis until ……., following my application to the Library of UNIWA and the approval from my supervisor.

The author
Stamatios-Michail Skaleris

I want to thank doctor Christos Zachariadis for providing me x-ray images and for assisting me in annotating the dataset. I would also like to thank my supervisor, Prof. Nelly Leligkou, for providing me guidance and feedback throughout my work and giving me new state-of-the-art ideas. Finally, I cannot forget to thank my family and friends for all the unconditional support in this academic journey.

Abstract

Early and accurate diagnosis of hip conditions, such as fractures and degenerative diseases, is crucial for ensuring that patients receive appropriate treatment on time. Delayed or incorrect diagnoses can lead to prolonged recovery times, worsened conditions, and higher risks of complications. In recent years, machine learning and deep learning have emerged as powerful tools for medical image analysis, offering the potential to assist healthcare professionals by automating parts of the diagnostic process. This thesis concerns the development of a multi-stage classification pipeline for the automated diagnosis of hip conditions from x-ray images, utilizing state-of-the-art deep learning techniques.

The dataset used in this study includes a combination of publicly available hip x-ray images and additional images provided by a physician, covering fractured, operated, and healthy hips, as well as hips with osteoarthritis. The classification pipeline consists of five stages, each addressing a specific diagnostic task. These stages include whether the image shows the left or right hip, whether the hip is normal or not, if an operation has been performed, the type of operation (arthroplasty or nailing), and the classification of fracture types and conditions (intertrochanteric fracture, subcapital fracture, osteoarthritis).

The final classification pipeline incorporates a ResNet50 model for the initial classification of left or right hip, achieving an accuracy of 89%. For the rest of the stages, VGG16 models were selected. The highest accuracy was obtained for classifying normal versus abnormal hips (98% recall) and the type of operation (100%). However, lower accuracy was observed in more complex tasks, such as differentiating between fracture types, where the model achieved an accuracy of 73%. Additionally, the classification where the hip is operated or not, the model achieved a recall of 91%. Transfer learning played a crucial role in boosting the performance of the pipeline, allowing the models to generalize well despite the limited availability of training data.

Despite the effectiveness of the proposed pipeline, several limitations were encountered. One of the main challenges was the limited availability of open-source medical imaging data, which hindered the training of more robust models. Additionally, hardware limitations restricted the ability to train larger models or explore more complex architectures. Future research can also utilize generative models to synthesize additional medical images, expanding the training dataset and improving model performance.

The findings of this thesis highlight the potential of deep learning techniques in automating medical diagnosis, particularly for hip-related conditions. Although automated diagnostic systems are still in the early stages of development, and should be used complementary to human expertise, they offer numerous benefits. These include faster and more efficient diagnosis, reduced diagnostic errors, and the ability to assist doctors in identifying additional areas of concern in medical images. Ultimately, automated systems could become valuable tools in healthcare, and drastically improve patient outcomes.

Keywords

Deep Learning, Transfer Learning, Automated Diagnosis, Medical Image Classification, Hip Fracture Detection

Περίληψη

Η έγκαιρη και ακριβής διάγνωση παθήσεων του ισχίου, όπως κατάγματα και εκφυλιστικές ασθένειες, είναι ζωτικής σημασίας για να διασφαλιστεί ότι οι ασθενείς λαμβάνουν την κατάλληλη θεραπεία έγκαιρα. Καθυστερημένες ή εσφαλμένες διαγνώσεις μπορεί να οδηγήσουν σε παρατεταμένους χρόνους ανάρρωσης, επιδείνωση των παθήσεων και υψηλότερους κινδύνους επιπλοκών. Τα τελευταία χρόνια, η μηχανική μάθηση και η βαθιά μάθηση έχουν αναδειχθεί ως ισχυρά εργαλεία για την ανάλυση ιατρικών εικόνων, προσφέροντας τη δυνατότητα να επωφεληθεί ο τομέας της υγείας, αυτοματοποιώντας μέρη της διαγνωστικής διαδικασίας. Η παρούσα διπλωματική εργασία αφορά την ανάπτυξη μιας σειράς αλγορίθμων ταξινόμησης πολλαπλών σταδίων για την αυτοματοποιημένη διάγνωση παθήσεων και χειρουργείων του ισχίου από εικόνες ακτίνων Χ, με τη χρήση τεχνικών βαθιάς μάθησης.

Το σύνολο δεδομένων που χρησιμοποιήθηκε σε αυτή τη μελέτη περιλαμβάνει έναν συνδυασμό διαθέσιμων στο κοινό εικόνων ακτινογραφίας ισχίου και εικόνων που παραχωρήθηκαν από έναν ορθοπεδικό χειρουργό, που καλύπτουν σπασμένα, χειρουργημένα και υγιή ισχία, καθώς και ισχία με οστεοαρθρίτιδα. Η διαδικασία της ταξινόμησης αποτελείται από πέντε στάδια, το καθένα για μια συγκεκριμένη διαγνωστική εργασία. Αυτά τα στάδια περιλαμβάνουν εάν η εικόνα δείχνει το αριστερό ή το δεξί ισχίο, εάν το ισχίο είναι φυσιολογικό ή όχι, εάν έχει γίνει επέμβαση, τον τύπο της επέμβασης (αρθροπλαστική ή ήλωση) και την ταξινόμηση των τύπων των καταγμάτων ή παθήσεων (διατροχαντήριο κάταγμα, υποκεφαλικό κάταγμα, οστεοαρθρίτιδα).

Το τελικό σύστημα ταξινόμησης ενσωματώνει ένα μοντέλο ResNet50 για την αρχική ταξινόμηση του αριστερού ή του δεξιού ισχίου, επιτυγχάνοντας ακρίβεια 89%. Για τα υπόλοιπα στάδια επιλέχθηκαν μοντέλα VGG16. Η υψηλότερη ακρίβεια επιτεύχθηκε για την ταξινόμηση των φυσιολογικών έναντι των μη φυσιολογικών ισχίων (98% recall) και του τύπου επέμβασης (100%). Ωστόσο, χαμηλότερη ακρίβεια παρατηρήθηκε σε πιο σύνθετες εργασίες, όπως η διαφοροποίηση μεταξύ των τύπων κατάγματος ή οστεοαρθρίτιδας, όπου το μοντέλο πέτυχε ακρίβεια 73%. Επιπρόσθετα, η ταξινόμηση όπου το ισχίο είτε έχει χειρουργηθεί είτε όχι, το μοντέλο πέτυχε recall 91%. Τα transfer learning μοντέλα έπαιξαν κρίσιμο ρόλο στην ενίσχυση της απόδοσης του συστήματος, επιτρέποντας στα μοντέλα να γενικεύουν παρά την περιορισμένη διαθεσιμότητα δεδομένων για εκπαίδευση.

Παρά την αποτελεσματικότητα του προτεινόμενου συστήματος, συναντήθηκαν αρκετοί περιορισμοί. Μία από τις κύριες προκλήσεις ήταν η περιορισμένη διαθεσιμότητα ιατρικών δεδομένων, η οποία εμπόδιζε την εκπαίδευση πιο ισχυρών μοντέλων. Επιπλέον, οι περιορισμοί hardware περιόρισαν τη δυνατότητα εκπαίδευσης βαθύτερων μοντέλων ή εξερεύνησης πιο περίπλοκων αρχιτεκτονικών. Μελλοντικές έρευνες μπορούν επίσης να χρησιμοποιήσουν μοντέλα για τη σύνθεση πρόσθετων ιατρικών εικόνων, επεκτείνοντας το σύνολο δεδομένων εκπαίδευσης και βελτιώνοντας την απόδοση του συστήματος.

Τα ευρήματα αυτής της εργασίας επισημαίνουν τις δυνατότητες των τεχνικών βαθιάς μάθησης στην αυτοματοποίηση της ιατρικής διάγνωσης, ιδιαίτερα για παθήσεις που σχετίζονται με την ορθοπεδική. Παρόλο που τα αυτοματοποιημένα διαγνωστικά συστήματα βρίσκονται ακόμα σε αρχικά στάδια ανάπτυξης και θα πρέπει να χρησιμοποιούνται συμπληρωματικά, προσφέρουν πολλά οφέλη. Αυτά περιλαμβάνουν ταχύτερη και πιο αποτελεσματική διάγνωση, μειωμένα διαγνωστικά σφάλματα και την ικανότητα να βοηθούν τους γιατρούς να εντοπίζουν πρόσθετα προβλήματα στις ιατρικές εικόνες. Τέλος, τα αυτοματοποιημένα συστήματα θα μπορούσαν να γίνουν πολύτιμα εργαλεία στον τομέα της υγείας και να βελτιώσουν δραστικά τα αποτελέσματα των ασθενών.

Λέξεις – κλειδιά

Deep Learning, Transfer Learning, Automated Diagnosis, Medical Image Classification, Hip Fracture Detection

Table of Contents

List of Tables

List of figures

Acronym Index

SGD: Stochastic Gradient Descent

ReLU: Rectified Linear Unit

AO/OTA: Arbeitsgemeinschaft für Osteosynthesefragen/Orthopaedic Trauma Association

AP: Anterior to Posterior

ORIF: Open Reduction and Internal Fixation

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

LSTM: Long Short-Term Memory

VGG: Visual Geometry Group

ResNet: Residual Network

DenseNet: Densely Connected Convolutional Network

MD: Medical Doctor

PA: Physician Assistants

FIF: Femoral Intertrochanteric Fracture

GAN: Generative Adversarial Network

GDPR: General Data Protection Regulation

INTRODUCTION

In recent years, hip fractures and the need for hip arthroplasty procedures have become significant public health concerns due to their increasing occurrences and the impact on individuals' quality of life. Accurate and timely detection of hip fractures and the need for arthroplasty plays a crucial role in improving patient recovery, locating the source of the condition, and reducing healthcare costs (Moran et al., n.d.). Manual diagnosis by medical professionals can be time-consuming, subjective, and prone to human error. Therefore, there is a need for automated and efficient methods to aid in the detection and classification of hip fractures and the subsequent requirement for hip arthroplasty.

Advancements in machine learning and artificial intelligence have shown great potential in various medical fields, improving disease detection and patient care. Leveraging the power of machine learning algorithms for hip fracture and hip arthroplasty detection offers an innovative approach to enhance accuracy and speed in clinical decision-making. By analyzing medical imaging data, such as x-rays, CT scans, or MRI scans, machine learning algorithms can identify subtle patterns, features, and abnormalities that may not be easily distinguishable by the human eye.

The subject of this thesis

This thesis scope is to develop a robust and reliable machine learning approach specifically tailored for the detection of hip abnormalities, or hip operations. The system is designed to process and analyze x-ray images. Through a comprehensive evaluation of various machine learning techniques and state-of-the-art algorithms, this work aims to identify the most suitable approach for accurate detection and classification of different hip abnormalities and hip operations.

The successful implementation of an automated machine learning algorithm for hip fracture and hip arthroplasty detection holds immense potential for healthcare providers, radiologists, and orthopedic surgeons. It can significantly reduce the time and effort required for diagnosis, enabling healthcare professionals to make informed treatment decisions promptly. Additionally, it has the potential to minimize the risk of misdiagnosis and unnecessary surgeries, thereby improving patient outcomes and overall healthcare efficiency.

Aim and objectives

Scope of this thesis is to contribute to the advancement of automated medical diagnosis by developing a robust system for hip abnormalities and hip operations detection. By leveraging deep learning techniques, this research strives to support clinical decision-making in orthopedic practice, ultimately improving patient care. Aim of this thesis is also to test various models' robustness in each classification task, proposing a pipeline that encompasses the best performing models. The pipeline architecture is designed to classify various hip fractures and diseases, covering a wider range of cases instead of focusing exclusively on one condition.

Methodology

The methodology adopted in this thesis involves an approach to classify hip x-ray images using a multi-stage deep learning pipeline. First, a dataset of x-ray images was compiled, including both publicly available images and additional data provided by Dr. Zachariadis, covering fractured, operated, and healthy hips. The classification task was broken down into multiple stages, each targeting a specific decision-making process: identifying the side (left or right hip), determining the state (normal or not normal), and further classifying between various medical conditions or operations (arthroplasty, nailing, different types of fractures, or osteoarthritis). For each stage, different deep learning models were trained and evaluated, with the most efficient models selected for the final pipeline based on their performance metrics. The final classification pipeline consists of sequentially applied models: ResNet50 for side classification, and VGG16 for the rest of the stages, which classify the hip's condition. Last but not least, the pipeline structure is designed to be time efficient. For example, in case a normal hip is detected, the process comes to an end, while it proceeds through further stages if abnormalities are present, with conditional logic.

Innovation

This work aims to pave the gap in relevant literature regarding the development of a pipeline that can combine various models that detect different cases and medical conditions. Thus, the proposed system will ultimately be capable of offering predictions that provide a more rounded diagnosis of the patient's state. Furthermore, the addition of medical image generation using a conditional Generative Adversarial Network (GAN) is being proposed to tackle the problem regarding the limitation in available data.

Structure

This thesis is structured into seven chapters, each providing information about different parts of the research process and background theory. Chapter 1 provides background information on relevant medical conditions, including various types of hip fractures and surgical procedures, establishing the clinical context for the work. Chapter 2 introduces the fundamental concepts of machine learning and deep learning, with an emphasis on common architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Chapter 3 reviews related work in the field, summarizing recent studies that use machine learning models for fracture detection. Chapter 4 focuses on the data used in this thesis, detailing the data collection process. Chapter 5 describes the methodology and experimental setup, including the training and evaluation of various models for each classification stage. Chapter 6 presents the final classification pipeline, explaining its construction and how the best-performing models were utilized. Finally, Chapter 7 includes the conclusions regarding the thesis, also providing information about existent limitations and future work.

1 Chapter 1: Background

Hip fractures and osteoarthritis are common and debilitating conditions, particularly in elderly populations, that require precise diagnosis and timely treatment to improve patient outcomes. These conditions not only result in significant pain and reduced mobility but also present challenges in terms of long-term recovery and quality of life. Understanding the types and treatments for these disorders is critical for developing effective diagnostic and therapeutic strategies. This chapter aims to provide a foundational understanding of hip fractures and

osteoarthritis, focusing on the importance of early detection and surgical interventions commonly employed to manage these conditions. By exploring the relevant medical background, this chapter sets the stage for understanding the role of automated diagnostic systems in improving clinical decision-making and patient care.

1.1 Importance of early detection of hip fractures

The hip is a ball-and-socket joint where the femur adjoins the ilium, ischium, and pubis of the pelvis. The femoral head is the ball and the acetabulum is the socket in this synovial joint. The proximal portion of the femur consists of the head, neck, and the greater and lesser trochanter (Figure 1). The greater trochanter, a bony prominence on the anterolateral surface of the proximal shaft of the femur, is the insertion site for the gluteus medius and gluteus minimus muscles. The lesser trochanter, a bony prominence on the proximal medial aspect of the femoral shaft, is the insertion site for the iliopsoas muscle (Ramponi et al., 2018).

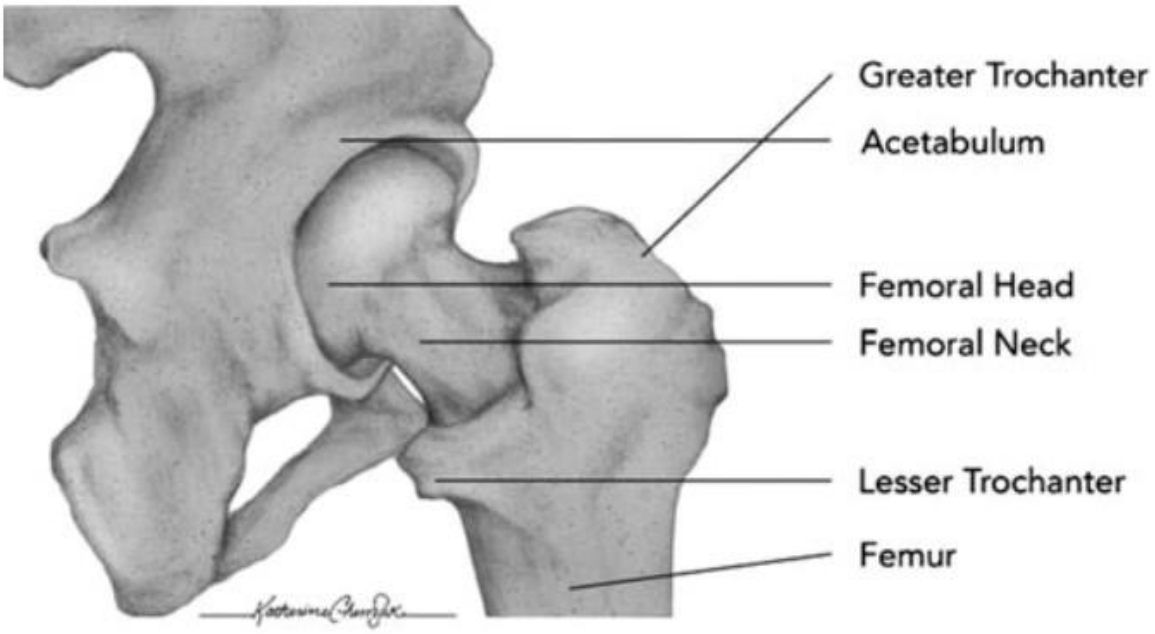


Figure 1. Anatomy of the hip bone (Ramponi et al., 2018).

Hip fracture is the most common major injury in the elderly and an important cause of mortality and morbidity (Moran et al., n.d.). Several research has been done over the years, some of them will be analyzed below.

1.1.1 Study conducted by Goldacre et al.

The following study was conducted by Goldacre et al. (Goldacre et al., 2002), and was published in 2002. This study examined emergency admissions of 8,148 individuals aged 65 and over with fractured neck of femur as the principal diagnosis. Out of these, 80.2% were women, with a mean age of 82.2 years. In the first month after fracture, the standardized mortality ratio was 1246 (95% confidence interval 1164 to 1331; general population 100). Adjusted standardized mortality ratios were 451 (397 to 509) in month 3, 238 (197 to 283) in month 6, and 187 (149 to 230) in month 12. Fractured femur was certified as the underlying cause in 16% of deaths within the first month and as a cause anywhere on the death certificate in 43%.

1.1.2 Study conducted by L. J. Melton, III

This research was conducted by L. J Melton (Melton, III, 1993), which was published in 1993. According to his study, hip fractures result in increased mortality and significant disability, often stemming from falls and osteoporosis, particularly affecting post-menopausal white women. Osteoporosis impacts one in four women in this demographic but affects fewer men and women of other races. In 1990, approximately 1.66 million hip fractures occurred globally, with half of them in Europe and North America. Despite this, there's considerable variation in hip fracture incidence rates within these regions, suggesting the influence of environmental factors that could be targeted to reduce hip fractures.

As stated in this research, the economic burden is substantial, especially in the United States, where a quarter of a million hip fractures annually incur costs exceeding $8 billion, primarily for acute medical care and nursing home services. Future costs are expected to rise due to global population aging, combined with the increasing hip fracture incidence rates in some areas. The elderly population is growing most rapidly in Asia, Latin America, the Middle East, and Africa, which are projected to contribute to over 70% of the anticipated 6.26 million hip fractures by 2050. Given the expensive nature of fracture treatment and the uncertainty of rehabilitation success, effective prophylaxis stands as the sole solution to mitigate the significant social and economic burden associated with hip fractures.

1.2 Basic hip fracture types

Hip fracture is the most common major injury in the elderly and an important cause of mortality and morbidity (Moran et al., n.d.). Hip fractures have many types, such as femoral neck fractures, intertrochanteric fractures, fracture of the greater trochanter, subtrochanteric fractures and femoral head fractures (Gray & Fischer, 2020). But according to Brunner et al. (Brunner et al., 2003), there are six basic fracture types:

1. Subcapital neck fracture
2. Transcervical neck fracture
3. Intertrochanteric fracture
4. Subtrochanteric fracture
5. Fracture of the greater trochanter
6. Fracture of the lesser trochanter

In this thesis, subcapital fractures and intertrochanteric fractures are contained in the classification system, which will be analyzed in more detail below.

According to Brunner et al. (Brunner et al., 2003), hip fractures split into two anatomic regions, intracapsular and extracapsular. Subcapital (femoral neck) fractures are included in the intracapsular fractures category, while intertrochanteric fractures and subtrochanteric fractures are included in the extracapsular fractures.

Anatomic Region	**Fracture**	**Frequency**	**Potential Complications**

Intracapsular	Subcapital (femoral neck) fracture	45% in the elderly; Male/Female ratio: 1:3	Avascular necrosis of the femoral head
Extracapsular	Intertrochanteric fracture	45% in the elderly; Male/Female ratio: 1:3	Rarely, malunion or nonunion; degenerative changes
Extracapsular	Subtrochanteric fracture	10%, with bimodal disruption (e.g. ages varying from 20 to 40 and above 60)	High rates of nonunion and implant (e.g. nails or devices implanted into the medullary cavity of the hip); High physical stress in the region may also cause fatigue

Table 1. Hip fractures classified based on general anatomic locations (Brunner et al., 2003).

Also, in certain cases of hip fractures, the fractured bone fails to heal or heals in a deformed position. These cases are called nonunion and malunion respectively (*Four Factors for Fracture Healing: Treatment of Nonunion and Malunion*, n.d.). Simply put, a malunion occurs when a fractured bone heals in a position that is not normal, which can cause reduced bone functionality. A nonunion instead, is the result of a fractured bone that fails to heal after a long time period (sometimes nine to twelve months) (*Malunion and Nonunion Fractures*, n.d.).

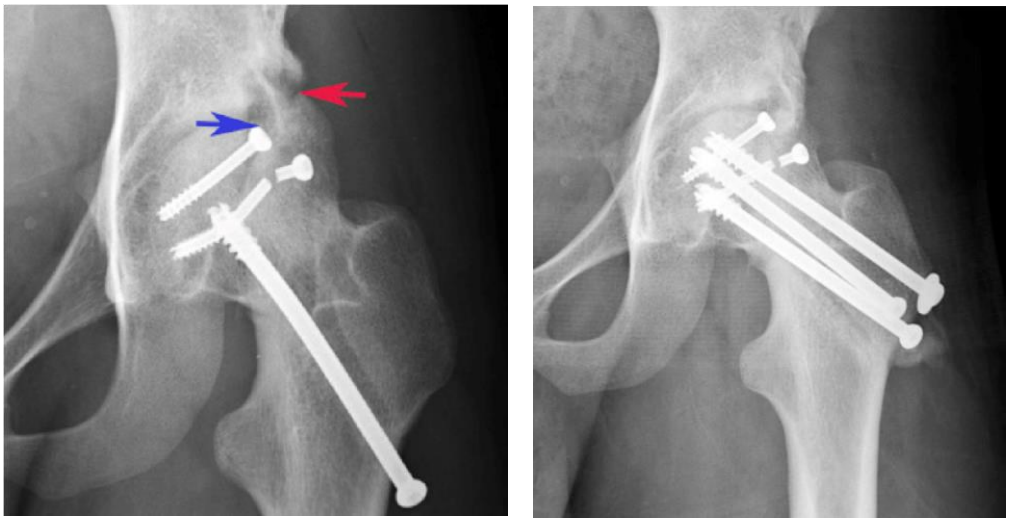


Image A. Preoperative anteroposterior x-ray, showing malunion of the femoral head (blue arrow).

Image B. Postoperative anteroposterior x-ray after arthroscopic osteosynthesis.

Figure 2. Preoperative and postoperative radiographs from a case of a femoral head malunion (Matsuda, 2014).

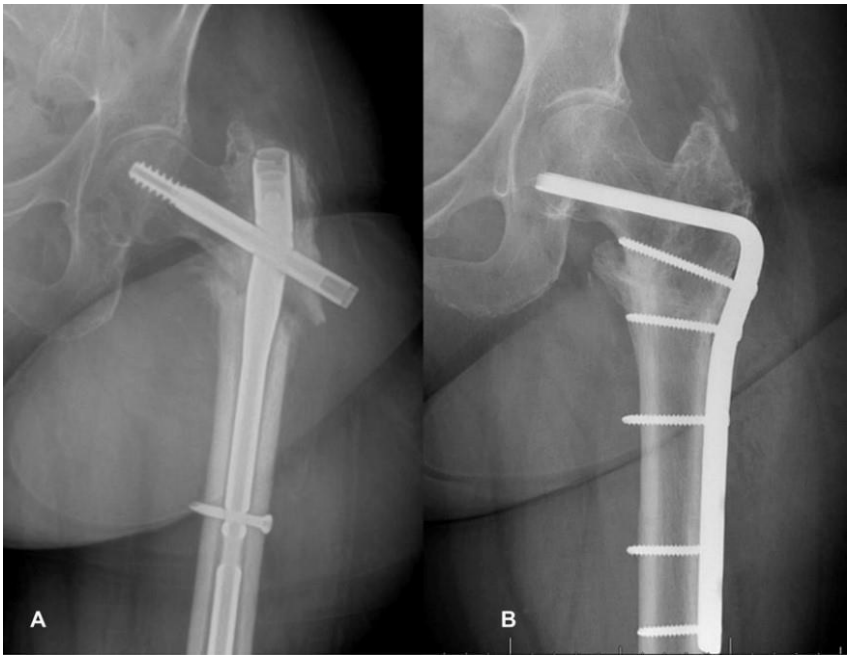


Figure 3. Patient with a nonunion of a femur fracture. **Image A** shows an anteroposterior radiograph, containing nonunion and hardware failure. **Image B** shows an anteroposterior radiograph healing, following nonunion repair (Egol et al., 2022).

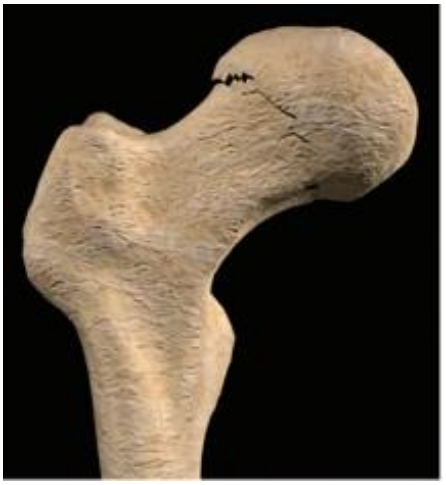
1.2.1 Subcapital/Femoral neck fracture

Subcapital fracture is the most common type of intracapsular neck of femur fracture. The fracture line extends through the junction of the head and neck of the femur (Shah, 2023). For classifying a subcapital fracture, many classification methods are proposed, but the Garden classification (Garden, 1961) and the Pauwel classification (Pauwels, 1965) are generally applied. These classification methods are preferred because these systems take into consideration the stability of a fracture.

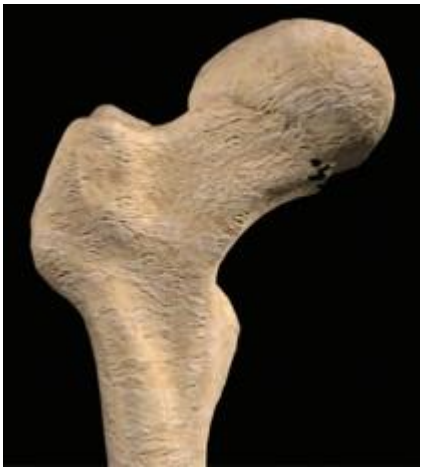
1.2.1.1 Garden Classification method

Garden classification (Garden, 1961) is based on the pre-reduction displacement of the femoral head. Furthermore, the displacement is graded as per the position of the principal compressive trabeculae. This system divides an intertrochanteric fracture into four types:

1) **Stage 1 (Figure 2A):** Subcapital fractures, which can be incomplete or valgus impacted (Sheehan et al., 2015) (humeral joint fragments impacted against the metaphyseal region, with separation of the tuberosities and minimal lateral deviation of the humeral head (Ribeiro et al., 2016)).
2) **Stage 2 (Figure 2B):** Fractures that are complete subcapital fractures but nondisplaced subcapital fractures (Sheehan et al., 2015).
3) **Stage 3 (Figure 2C):** Fractures that are complete subcapital fractures that are partially displaced (Sheehan et al., 2015).
4) **Stage 4 (Figure 2D):** Fractures that are complete subcapital fractures that are fully displaced (Sheehan et al., 2015).

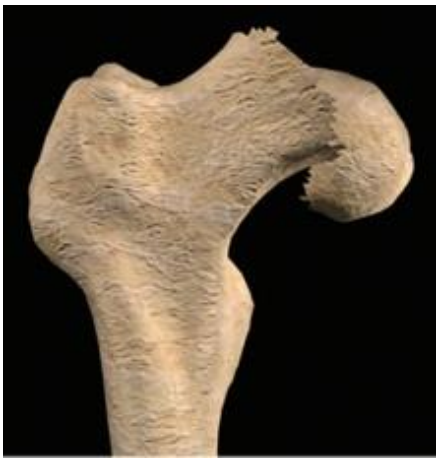


B. Stage 1 subcapital neck fracture



B. Stage 2 subcapital neck fracture

B. Stage 3 subcapital neck fracture



B. Stage 4 subcapital neck fracture

Figure 4. The Garden classification system for subcapital femoral neck fractures (Sheehan et al., 2015).

1.2.1.2 Pauwel Classification method

Pauwel classification (Pauwels, 1965) is based on the post-reduction angulation of the fracture line to the horizontal line, evaluated on an anterior to posterior (AP) radiograph (Shah, 2023). This system is divided into three types, based on the angle of the fracture relative to the horizontal plane (Figure 5):

1) **Degree I:** Angle relative to horizontal pane < 30°.
2) **Degree II:** 30° < Angle relative to horizontal pane < 50°.
3) **Degree III:** 50° < Angle relative to horizontal pane.

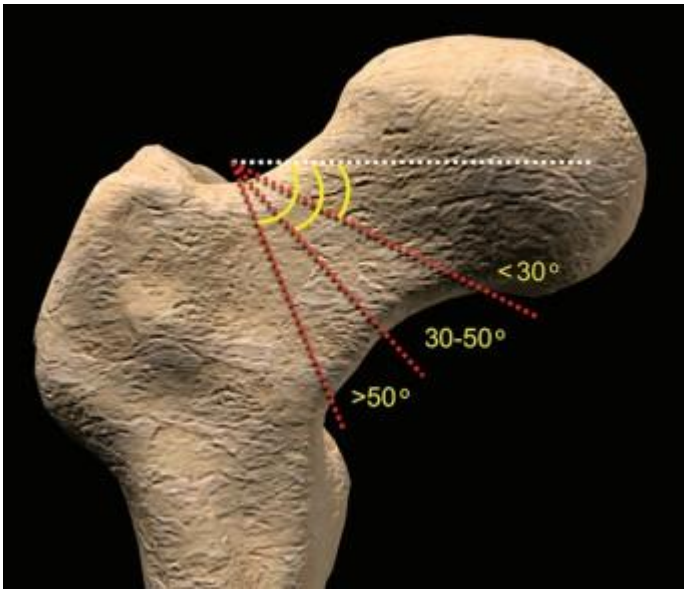


Figure 5. Pauwels classification system for postreduction femoral neck fractures, determined by the angle of the fracture relative to the horizontal plane (white line) (Sheehan et al., 2015).

1.2.2 Intertrochanteric fracture

Intertrochanteric fractures are present in the region between the greater and lesser trochanters. Because they occur in the furthest anatomic regions of the hip joint capsule, they are classified as extracapsular fractures. The cancellous bone (characterized by its spongy, porous structure) is well vascularized, meaning that rarely a nonunion or osteonecrosis will arise, which make the healing of the fracture a lot more complicated (Koval & Zuckerman, 2013).

1.2.2.1 Classification of intertrochanteric fractures

The classification of intertrochanteric fractures is a lot more complicated when choosing a classification system to follow. Extensive research has been conducted over the years to scrutinize the reasons behind poor reliability and reproducibility of fracture classifications. Unfortunately, the challenges associated with the reliability of classification have led to loss of enthusiasm for the classification process. The demanding nature of this process has often been overlooked in favor of more popular and commonly employed classification systems (Marsh et al., 2007).

In the study of Yıldırım et al. (Yıldırım et al., 2022), they aimed to evaluate the reliability for five classification systems:

- Boyd-Griffin classification system
- Evans/Jensen classification system
- Evans classification system
- AO/OTA (Arbeitsgemeinschaft für Osteosynthesefragen/Orthopaedic Trauma Association) (main and subgroups) classification system

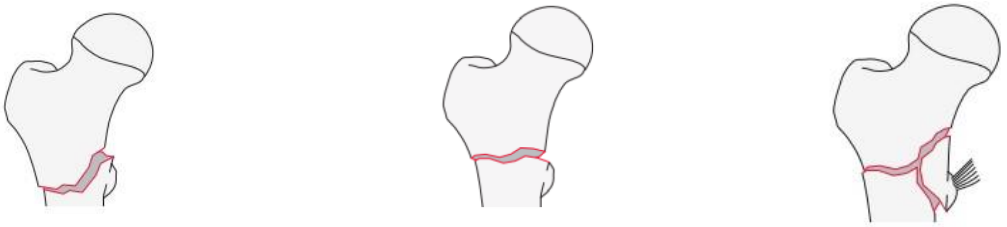
- Tronzo classification system

Radiological images from sixty patients (13 males and 47 females, with ages ranging from 61 to 96 years) were evaluated and classified by five residents, five orthopedics and five traumatology surgeons according to the aforementioned classification systems. Intraobserver and interobserver reliability were calculated using Cohen's κ-coefficient (Cohen, 1960). Kappa measures the degree of agreement between a pair of variables, frequently used as a metric of interrater agreement, particularly in situations involving judgments rather than measurement. Kappa values range from [-1,1], with 1 indicating complete agreement and 0 no agreement or independence. According to a study of Yinglin (Yinglin, 2020), the standard for an acceptable kappa value is arbitrary. According to Fleiss' arbitrary guidelines, which paper is often referenced, 0.75 is considered as excellent (Fleiss, 1971). However, it's essential to note that kappa is intrinsically nonlinear, is not adept at handling errors and retains bias influence. The classification that has the best harmony both among residents and surgeons, and between residents and surgeons is the OTA main group classification. The results can be seen in Table 2.

	Resident group intraobserver (95% Confidence Interval)	Surgeon group intraobserver (95% Confidence Interval)	Resident-Surgeon group interobserver (95% Confidence Interval)
Boyd-Griffin	0.660 (0.550-0.770)	0.658 (0.550-0.770)	0.572 (0.532-0.616)
Evans-Jensen	0.625 (0.600-0.655)	0.484 (0.434-0.542)	0.498 (0.450-0.553)
Evans	0.557 (0.519-0.595)	0.456 (0.409-0.053)	0.438 (0.400-0.481)
AO/OTA main group	0.744 (0.708-0.785)	0.741 (0.696-0.797)	0.699 (0.649-0.750)
AO/OTA subgroup	0.516 (0.498-0.540)	0.488 (0.418-0.558)	0.444 (0.418-0.470)
Tronzo	0.528 (0.501-0.592)	0.529 (0.489-0.569)	0.554 (0.506-0.614)

Table 2. Intraobserver kappa values of resident group and surgeon group, and interobserver kappa value of resident-surgeon evaluations (Yıldırım et al., 2022).

The AO/OTA classification method can be seen in Figure 6 below.



Subgroup: Simple oblique fracture

Subgroup: Simple transverse fracture

Subgroup: Wedge or multifragmentary fracture

Figure 6. AO/OTA classification method for intertrochanteric (reverse obliquity) fractures (Meinberg et al., 2018).

1.2.3 Osteoarthritis

The most prevalent joint disorder in the United States is osteoarthritis, which is the most common cause of disability in the elderly, with approximately 200,000 total hip replacements performed each year. Radiographs that contain osteoarthritis of the hip occurs in about 5% of the population over the age of 65 years (Lane, 2007). The difficult part of classifying an occurrence of osteoarthritis is that not all patients present symptoms. The inconsistency between changes in radiograph images and symptoms may account for false negative or false positive findings in well-established studies of osteoarthritis of the hip.

The term “osteoarthritis” is used to represent a hypernym group of joint disorders, presenting joint pain and stiffness. Also, the pathogenesis of osteoarthritis is not completely understood. In most cases, osteoarthritis most likely starts with degradation of the articular cartilage in a localized, nonuniform manner. In the following period, a subsequent thickening of the subchondral bone occurs, new bony outgrowths at joint margins (osteophytes), and mild-to-moderate synovial inflammation. The events that initiate osteoarthritis are not clearly defined, but are probably due to abnormal signals that alter the chondrocyte phenotype so that it synthesizes proteins that degrade the matrix and causes degeneration of the joint.

There are two main categories of osteoarthritis of the hip, primary (idiopathic) or secondary (systemic or localized) disease. Risk factors for primary osteoarthritis of the hip include:

- Old age
- Genetic predisposition for the disease
- High bone mass
- Increased body mass index
- Participation in weight-bearing sports
- Occupations that require prolonged standing, lifting, or moving heavy objects.

Risk factors for the secondary causes (systemic) include:

- Hemochromatosis
- Hypothyroidism
- Hyperparathyroidism
- Hyperlaxity syndromes
- Acromegaly
- Paget's disease ● Chondrocalcinosis ● Gout.

Also, localized risk factors include:

- Joint injury
- Legg–Calvé–Perthes disease
- Developmental deformities
- Osteonecrosis
- Acetabular dysplasia
- Rheumatoid or septic arthritis as a result of cartilage damage.

Finally, signaling pathways and polymorphisms in combination with the development and metabolism of bone and cartilage, are also linked with the risk of developing osteoarthritis (Lane, 2007).

1.2.4 Surgical Treatment: arthroplasty, nailing

In this section, information will be provided regarding the different treatments that need to be offered to patients with subcapital fractures or osteoarthritis, highlighting the medical decision-making process and how these conditions are managed through surgical interventions.

1.2.4.1 Treatment of Subcapital Fractures

Treatment of subcapital neck fractures fall into two categories: internal fixation or arthroplasty (hemiarthroplasty or total hip arthroplasty). Instead of choosing a treatment option based on a diagnosis-related approach, treatment options now also take into consideration the patient's age, functional demands and the individual's risk profile. For younger patients, the treatment follows with urgent open reduction and internal fixation (ORIF) surgery, with the goal of anatomic reduction. Anatomical reduction is the alignment of the fractured bone fragments, with the aim to reconstruct the broken bone as closely as possible to its original form, ensuring optimal healing and restoration of function to the affected bone and surrounding joints (Dogramadzi et al., 2014). For elderly patients, their cognitive function should be determined. For cognitive functional patients, the best approach is a total hip arthroplasty. Meanwhile, for cognitive dysfunctional patients, a bipolar hemiarthroplasty or a total hip arthroplasty with the use of larger heads and/or constrained sockets are a feasible option (Callaghan et al., 2012).

1.2.4.2 Treatment of Intertrochanteric Fractures

Intertrochanteric fractures are typically treated surgically, as nonoperative treatment is generally reserved for non-ambulatory patients or those with a high risk of perioperative mortality. Non-surgical treatment carries significant risks, including pneumonia, urinary tract infections, pressure ulcers, and deep vein thrombosis.

Therefore, surgery is the preferred treatment for most patients to restore mobility and minimize complications. The choice of surgical method depends on the fracture pattern and stability, as the type of implant used directly affects the success of the treatment. Fractures involving the lateral femoral wall, or unstable patterns with comminution or subtrochanteric extension, are often treated with intramedullary nailing, as this method provides superior stability compared to sliding hip screws.

For stable intertrochanteric fractures, a sliding hip screw may be used, especially when the lateral femoral wall remains intact. This technique offers comparable outcomes to intramedullary nailing in such cases, and its advantages include dynamic interfragmentary compression and lower cost. However, it has drawbacks, including higher blood loss and the need for an open surgical approach. Intramedullary nailing, on the other hand, is preferred for unstable fracture patterns, as it is minimally invasive and reduces blood loss. Arthroplasty is rarely used for intertrochanteric fractures and is typically reserved for complex cases, such as severely comminuted fractures or when internal fixation is not feasible due to osteoporotic bone or pre-existing degenerative conditions (Attum & Pilson, 2024).

1.2.4.3 Treatment of Osteoarthritis

Treatment of osteoarthritis has two main goals; relieving the patient from pain and maintaining functionality. As analyzed in the paper published by Lane et. al (Lane, 2007), there are several ways for treating osteoarthritis, such as nonpharmacologic treatment (balance improvement when walking with a cane, self-help education classes etc), pharmacologic treatment (mainly drugs to manage pain and/or inflammation) and a surgical approach.

Regarding surgical approaches, total hip arthroplasty is an effective treatment for patients suffering from chronic pain and functional impairment. Also, a rehabilitation program may follow for several months, to regain a reasonable functionality of the operated hip joint. Maximal pain relief and improvement in functions may take up to 12 months. Other effective approaches are resurfacing arthroplasty (capping the femoral head and preserving bone of the proximal femur (Mont et al., 2006)) and osteotomy (i.e. proximal femoral osteotomy (Tannast & Siebenrock, 2009)).

2 Chapter 2: Machine Learning

Machine learning is a subset of artificial intelligence that enables computers to learn from experience and improve their performance over time without being explicitly programmed. It involves algorithms that can analyze data, recognize patterns, and make decisions with minimal human intervention.

2.1 Machine Learning in general

The concept of machine learning was first introduced by Arthur Samuel in 1959, who described it as a field of study that gives computers the ability to learn without being explicitly programmed (Samuel, 1959).

A machine learning algorithm uses input data to achieve a desired task for the purpose of producing a particular outcome. These algorithms automatically adjust their configuration

through repetition to improve their performance in a given task. This adaptive process, known as training, involves providing input data samples and desired outcomes. The algorithm optimizes its configuration to not only achieve good performance with training data, but also to generalize and perform well with new, unseen data. This continuous learning process allows a proficient algorithm to refine its capabilities over time by processing new data and learning from mistakes (El Naqa et al., 2015).

The field of machine learning is multi-disciplinary, having a wide-range of research fields, i.e. psychology, neuroscience, information theory, and computational complexity theory. It has a wide range of applications, including email spam filtering, fraud detection on social networks, online stock trading, medical diagnosis, and self-driving cars. Machine learning algorithms are designed to handle complex real-world problems and can be categorized into different paradigms (Alzubi et al., 2018).

2.1.1 Machine Learning Paradigms

Three main categories of machine learning are supervised learning, unsupervised learning and reinforcement learning. All of them are being briefly presented in the subsections below.

2.1.1.1 Supervised Learning

Supervised learning is a type of machine learning where the model is trained on labeled data. It involves using known input-output pairs to enable the algorithm to learn and make predictions. The process consists of two main steps: training, where the model learns from the data, and testing, where the model's predictions are evaluated, on data different from that of the training process. The goal is to map input data to known output labels so that when the model encounters new, unseen data, it can accurately predict the corresponding output. Figure 7 explains this concept.

In essence, supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. Each example is a pair consisting of an input object (typically a vector) and a desired output value. Supervised learning algorithms analyze the training data and produce an inferred function, which can be used for mapping new examples. Supervised learning can be further divided into classification tasks, where the output is a discrete label, and regression tasks, where the output is a continuous value. These algorithms are widely used for predictive tasks or future event forecasting (Preeti & Dhankar, 2017).

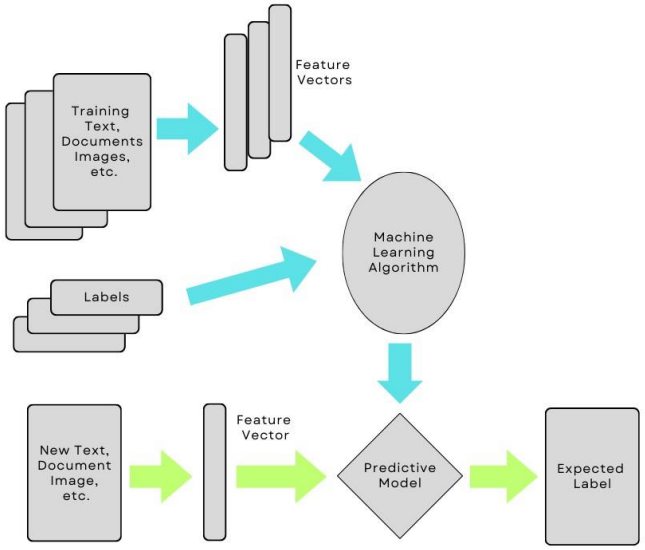


Figure 7. Supervised Learning (Preeti & Dhankar, 2017)

More specifically, during the training process, the machine generates a vector of scores for each given image, each score corresponding to a category. Ideally, the correct category should have the highest score. An objective function is used to measure the error between the produced scores and the desired scores. The machine then adjusts its internal parameters, known as weights, to minimize this error. These weights are real numbers that shape the machine's input-output function. In deep learning systems, there can be hundreds of millions of weights and training examples. To fine-tune the weights, the learning algorithm calculates a gradient vector that shows how the error changes with small adjustments to each weight. The weights are then updated in the opposite direction of this gradient.

In practice, most practitioners use a method called stochastic gradient descent (SGD). This involves presenting the input vector with a few examples, calculating the outputs and errors, determining the average gradient for those examples, and then adjusting the weights accordingly. This process is repeated with many small sets of examples from the training set until the average objective function stops decreasing. It is called stochastic because each small set of examples provides a noisy estimate of the average gradient over all examples. This procedure usually finds a good set of weights quickly compared to more complex optimization techniques. After training, the system's performance is evaluated on a different set of examples called a test set, which assesses the machine's generalization ability to produce sensible answers on new, unseen inputs (LeCun, 2015).

2.1.1.2 Unsupervised Learning

Unsupervised learning is a type of machine learning that operates on data without predefined labels, aiming to identify underlying patterns or structures within the dataset. It’s particularly useful for exploratory data analysis, such as customer segmentation in marketing campaigns. Unlike supervised learning, which relies on labeled input-output pairs for training,

unsupervised learning algorithms infer the natural grouping or structure of the data based on intrinsic characteristics. Figure 8 explains this concept.

One common application of unsupervised learning is clustering, where the algorithm organizes data into groups based on similarities. This can be applied to various fields, including bioinformatics and image compression. Techniques like k-means and hierarchical clustering are popular methods within this domain. Unsupervised learning is also adept at dimensionality reduction, helping to simplify datasets by reducing the number of variables under consideration, which in turn can enhance the performance of other machine learning algorithms (Preeti & Dhankar, 2017).

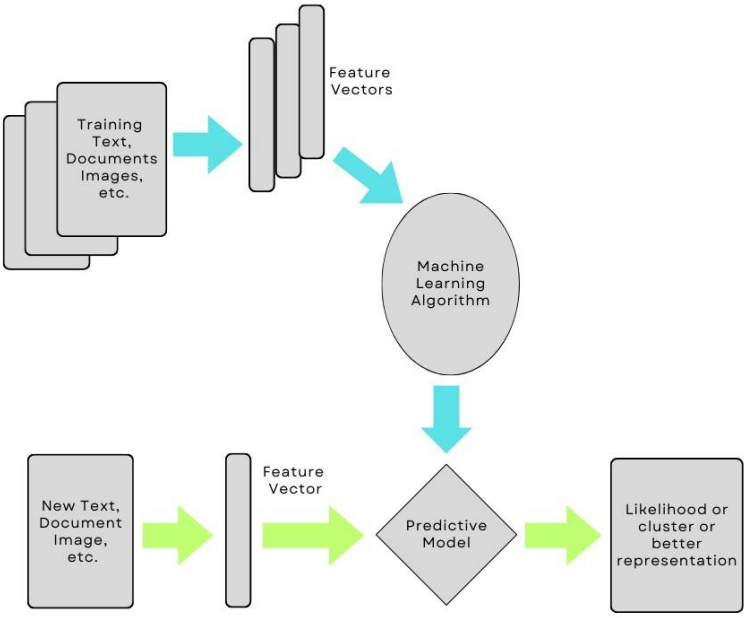


Figure 8. Unsupervised Learning (Preeti & Dhankar, 2017).

2.1.1.3 Reinforcement Learning

Reinforcement learning is a computational approach where an agent learns to make decisions by trial and error, receiving feedback from its actions in the form of rewards or penalties. This method combines elements from psychology, engineering, and artificial intelligence, allowing the agent to develop a strategy that maximizes its long-term gains from a specific task. The agent's objective is to accumulate the highest possible amount of reward, which is defined by a reward function that dictates what is beneficial for the agent within its environment.

The core concept of reinforcement learning involves the agent's interactions with its environment, where it performs actions and observes the outcomes to adjust its behavior. Key components include the reward function (r_t), state (s_t) and action (a_t) definitions, and the

policy function, which maps states to actions. Over time, the agent refines its policy based on the results of its actions, aiming to improve the expected return. Figure 9 explains this concept. This learning process is dynamic, with the agent continually updating its value function, which

estimates the long-term benefits of its actions, leading to more informed decisions and better performance in the task at hand (Sutton & Barto, 1999).

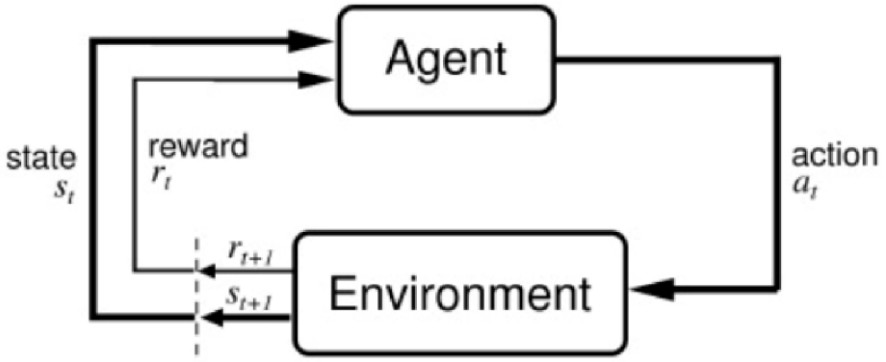


Figure 9. Reinforcement Learning (Whiteson, 2010).

2.2 Deep Learning

Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep learning methods are representation learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned (LeCun, 2015, 1).

2.2.1 Neural Networks

Neural network architectures consist of a multilayer stack of modules. Most of the modules undergo learning, while many of them perform non-linear input-output mappings. Each module transforms its input, in order to improve representations. This can be achieved by enhancing both the selectivity and invariance of the representation. Using several non-linear layers, a system can execute highly complex functions of its inputs, focusing on meaningful changes. This means that the model is capable of being sensitive to details of interest, while remaining unaffected by significant irrelevant variations such as the lighting, surrounding objects and general background (LeCun, 2015).

Multilayer architectures are trained using stochastic gradient descent. Provided the modules are relatively smooth functions of their inputs and internal weights, backpropagation can compute the gradients. To elaborate, backpropagation calculates the gradient of an objective function relative to the weights of a multilayer module stack by applying the chain rule for derivatives. The idea is that the gradient of the objective, with respect to a module's input, can be derived by tracing back from the gradient concerning the module's output (or the input of the next module). This backpropagation equation can propagate gradients through all modules, starting from the output (where the network makes its predictions) down to the bottom (where the external input enters). Once these gradients are available, computing gradients concerning each module's weights becomes straightforward. Multilayer neural networks and backpropagation can be seen in more detail in Figure 10.

In many deep learning applications, feedforward neural network architectures are applied, which map a fixed-size input (like an image) to a fixed-size output (such as probabilities for each

label/category). Transitioning from one layer to the next involves units computing a weighted sum of their inputs from the previous layer and applying a non-linear function. The most widely used non-linear function is the Rectified Linear Unit (ReLU), defined as f(z) = max(z, 0) (Hara et al., 2015). Previously, neural networks used smoother non-linearities like tanh(z) or 1/(1+exp(−z)), but ReLU generally facilitates faster learning in neural networks, allowing for deep supervised network training without unsupervised pre-training. Units not in the input or output layer are known as hidden units. These hidden layers transform the input non-linearly, making categories linearly separable by the final layer.

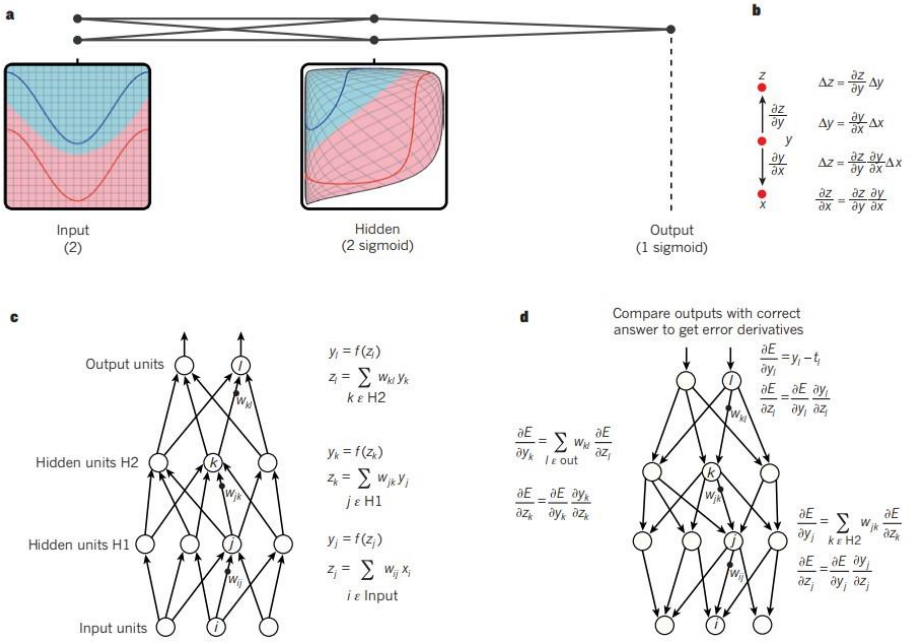


Figure 10. Multilayer neural networks and backpropagation. (LeCun, 2015) **a)** A multilayer neural network can transform input space to make data classes linearly separable. This example uses two input units, two hidden units and one output unit, but networks for tasks like object recognition or natural language processing (NLP) can contain tens or hundreds of thousands of units. **b)** The chain rule of derivatives explains how two small effects combine. A small change Δx in x is first transformed into a small change Δy in y by multiplying with ∂y/∂x (the partial derivative). The change Δy results in a change Δz in z. Substituting one equation into the other provides the chain rule of derivatives: how Δx turns into Δz by multiplying by the product of ∂y/∂x and ∂z/∂x. This also applies when x, y and z are vectors, and the derivatives are Jacobian matrices. **c)** The equations for the forward pass in a neural network with two hidden layers and one output layer involve computing the total input z to each unit as a weighted sum of the outputs from the layer, then applying a non-linear function f(.) is applied to get well as more traditional sigmoids, like the hyberbolic tangent $f(z) = max(0, z)$ the output. Common non-linear functions in neural networks include ReLU , as
and the logistic function
. **d)** The equations for the backward pass compute the error derivative
with$ff((zz))$ respect== $(1exp/(1$to$($ $z$each$+) - exp$unit's
$exp(-($output$-z))z))/$at$(exp$each$($zhidden$) + exp$layer.$(-$ zIt$))$is a weighted sum of the error derivatives

concerning the total inputs of the units in the layer above. The error derivative with respect to the output, is then converted into the error derivative with respect to the input, by multiplying by the gradient of $f(z)$. At the output layer, the$y$error$_l$ $-$ $t$derivative$_l$ with respect to the output, is derived by$_2$ differentiating the cost function, giving if the cost function for unit l is $0.5(y_l - t_l)$, where

t_l is the target value. Once the $\partial E/\partial z_k$ is known, the error derivative for the weight w_{jk} on the connection from unit j in the lower layer right is $y_j \partial E/\partial z_k$.

2.2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specialized neural networks designed to process data that is organized as multiple arrays, such as 2D images (e.g. a colored image composed of 3 2-Dimensional arrays, containing pixel intensities in the three color channels, Red Green and Blue (RGB)), 1D signals (e.g. language) or 3D videos. They leverage four fundamental principles that utilize the properties of natural signals: local connections, shared weights, pooling, and deep layering. The architecture of ConvNets is typically structured in stages, with the starting stages consisting of convolutional and pooling layers (Figure 11). In each convolutional layer, units are organized in feature maps, where each unit is connected to local patches in the previous layer's feature maps through a set of weights, called a filter bank. Then, the result of this sum, called a local weighted sum, passes through a non-linearity (e.g. ReLU). This connection allows for the detection of local patterns, with all units sharing the same filter bank and each feature map using different filter banks, to identify various patterns across the data. The reason for the CNNs' architecture is based on two factors:

1. In data that consist of arrays (e.g. images), it is common for local groups of values to be greatly correlated, thus creating distinctive local motifs that can be detected easily.
2. Local statistical data of images as well as other signals are invariant to location. This means that if a motif is present in one part of an image, it could appear anywhere else as well. This is the reason why units at different locations share the same weights and detect the same patterns in different parts of an array, as mentioned above. The filtering operation performed by a feature map is a discrete convolution (LeCun, 2015).

To conclude, the convolutional layer's primary function is to identify local combinations of features from the preceding layer.

Moreover, the pooling layer's role is to combine semantically similar features into one. Since the relative positions of features within a motif can vary, reliably detecting the motif is achieved by

generalizing the position of each feature. Typically, a pooling unit determines the maximum value within a local patch of units in a feature map, or sometimes across several feature maps. Adjacent pooling units receive input from patches that are offset by more than one row or column, which reduces the representation's dimensionality and generates invariance to small positional shifts and distortions. Multiple stages of convolution, pooling and non-linearity are stacked, followed by additional convolutional and fully-connected layers. Gradient backpropagation through a ConvNet is straightforward, like in a regular deep network, enabling the training of all weights in the filter banks across the network.

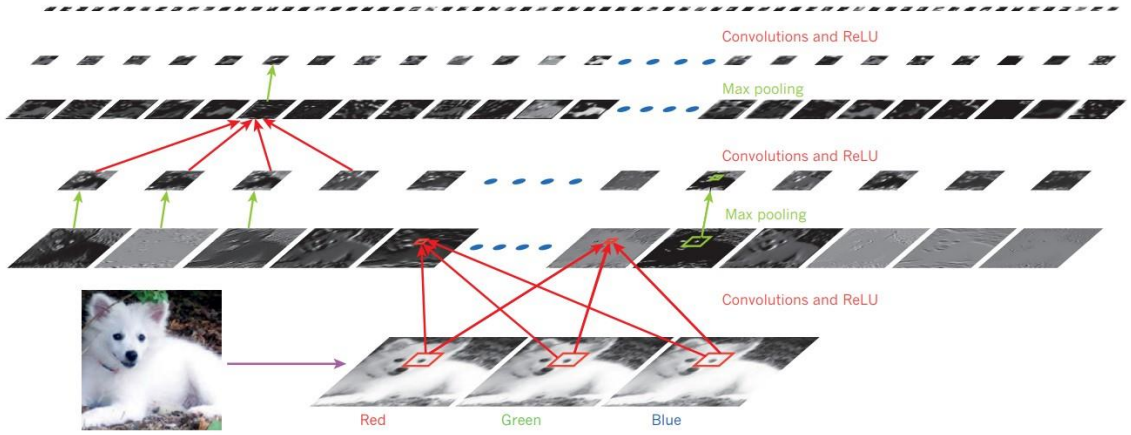


Figure 11. Convolutional Network typical architecture. The outputs from each layer of a standard convolutional network architecture applied to a dog's image. Each image represents a feature map corresponding to one of the learned features, identified across different positions in the image. Information moves from the bottom up, with lower-level features serving as oriented edge detectors, and a score is calculated for each class in the output (LeCun, 2015).

2.2.1.2 Recurrent Neural Networks

For tasks involving sequential inputs, such as language and speech, recurrent neural networks (RNNs) are most times more effective (Figure 11). RNNs process a sequence one element at a time while maintaining a "state vector" in their hidden units, which implicitly stores information about the history of the sequence. RNNs are powerful dynamic systems, but training them has been challenging due to the tendency of backpropagated gradients to either explode or vanish over time. However, advancements in architecture and training methods have improved RNN performance, making them effective for tasks like predicting the next character or word in a sequence, as well as more complex tasks.

For instance, an English "encoder" network can be trained to convert an English sentence into a "thought vector" by processing it one word at a time. This vector can then initialize a French "decoder" network, which generates a probability distribution for the first word of the French translation. The process continues, with the decoder producing distributions for subsequent words, ultimately generating a French sentence based on the English input. This approach to machine translation has quickly become competitive with state-of-the-art methods, challenging the idea that understanding a sentence requires internal symbolic expressions and suggesting that reasoning may involve many simultaneous analogies contributing to a conclusion (LeCun, 2015).

Beyond translating between languages, the same concept can be applied to translating the meaning of an image into a sentence. In this case, a deep ConvNet acts as the encoder, converting pixel data into a representation vector, while an RNN functions as the decoder, similar to those used in machine translation and neural language modeling.

When unfolded in time, RNNs resemble deep feedforward networks with shared weights across all layers. Despite their design to learn long-term dependencies, it is theoretically and empirically challenging for them to retain information over extended periods. To address this, Long Short-Term Memory (LSTM) networks were introduced, featuring special hidden units that naturally remember inputs for extended durations. These units, known as memory cells, accumulate external signals and decide when to clear the memory. LSTM networks have proven more effective than traditional RNNs, particularly when multiple layers are used for each time step. They are capable of powering entire speech recognition systems, from acoustics to transcription. Additionally, LSTM networks that are related to them are now commonly used in encoder and decoder networks that demonstrate powerful performance in machine translation tasks. Memory networks have also shown excellent performance on standard question-answering benchmarks, where they use memory to retain the story that the network will later be questioned about (LeCun, 2015).

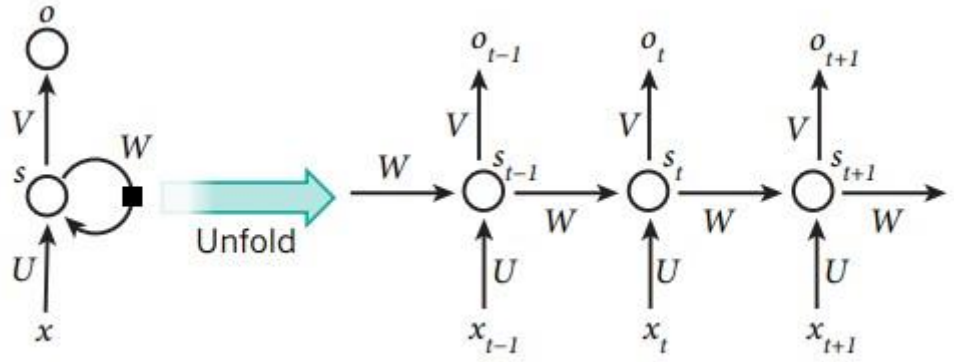


Figure 12. Recurrent Neural Network (RNN) architecture. A Recurrent Neural Network typical architecture that unfolds over time during its forward computation. The artificial neurons, such as hidden units grouped under node s with values s_t at time t, receive inputs from other neurons at previous time steps. This is indicated by the black square on the left, representing a delay of one time step. This allows a recurrent neural network to map an input sequence, with elements x_t, to an output sequence, with elements o_t, where each o_t depends on all previous x_t'(for t′ ≤ t). The same parameters version where$U, V,$the$W$ network generates a sequence of outputs (e.g., words), with each output used as the (matrices) are reused at each time step. Various other architectures are possible, including a
input for the next time step. The backpropagation algorithm (Fig.10) can be directly applied to the computational graph of the unfolded network, shown on the right, to compute the derivative of a total error (such as the log-probability of generating the correct output sequence) with respect to all the states s_t and all the parameters (LeCun, 2015).

2.3 Transfer Learning Models

Transfer learning is a rapidly growing topic that has the potential to drive the success of machine learning both in research and industry. It is particularly useful when there is a lack of data for specific tasks, as collecting and labeling data can be costly and time-consuming. Also, recent privacy concerns make it challenging to use real data from users. Transfer learning enables the quick prototyping of new machine learning models, by leveraging pre-trained models from a source task, avoiding the need to train on millions of images, which takes a lot of time and requires GPUs that have a very high cost (Ribani & Marengoni, 2019).

2.3.1 Visual Geometry Group (VGG16)

VGG16 is a convolutional neural network (CNN) architecture that was proposed by Karen Simonyan and Andrew Zisserman from the University of Oxford in their 2014 paper titled "Very Deep Convolutional Networks for Large-Scale Image Recognition" (Simonyan & Zisserman, 2014). The name “VGG16” comes from the Visual Geometry Group (VGG) at Oxford, and the “16” refers to the 16 weight layers in the network.

The VGG16 architecture is known for its simplicity and uniformity, using small 3x3 convolution filters throughout the network. Below is a detailed breakdown of its architecture:

- **Input Layer:** The input to the network is a fixed-size 224x224 RGB image. The only preprocessing done is subtracting the mean RGB value, computed on the training set, from each pixel.
- **Convolutional Layers:** The network has 13 convolutional layers. These layers use very small receptive fields: 3x3 (which is the smallest size to capture the notion of left/right, up/down, center). The convolution stride is fixed to 1 pixel, and the spatial padding of the convolution layer input is such that the spatial resolution is preserved after the convolution.
- **Max-Pooling Layers:** There are five max-pooling layers, each following some of the convolutional layers. Max-pooling is performed over a 2x2 pixel window, with stride 2.
- **Fully Connected Layers:** The stack of convolutional layers is followed by three fully connected layers. The first two fully connected layers have 4096 channels each. The third fully connected layer performs 1000-way ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) classification and thus contains 1000 channels (one for each class).
- **Activation Function:** All hidden layers are equipped with the ReLU function. The final layer is a softmax layer.

A representation of the above can be also seen in Figure 13 below.

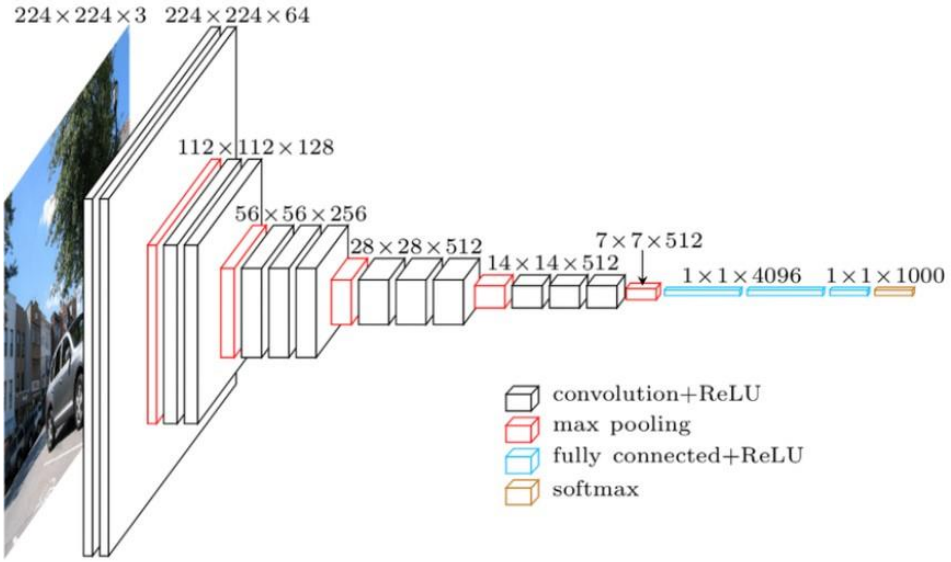


Figure 13. VGG16 neural network architecture (Cano, n.d.).

The use of small 3x3 filters throughout the network makes the architecture simple and uniform. Additionally, the depth of the network allows it to learn complex features and achieve high accuracy on large-scale image recognition tasks. VGG16 achieved state-of-the-art results on the ImageNet dataset and has been widely used as a backbone for many other computer vision tasks. VGG16's architecture has been influential in the development of deeper and more complex neural networks, demonstrating the importance of depth in achieving high performance in image recognition tasks.

2.3.2 Residual Network (ResNet50)

ResNet50 is a deep convolutional neural network that was introduced by He et al. in 2015 (He et al., 2015). ResNet50 is part of the Residual Networks (ResNet) family. It is designed to ease the training of very deep networks by introducing residual learning.

ResNet50 uses residual blocks, which help in training deeper networks by allowing gradients to flow through the network more effectively. Each block includes shortcut connections that skip one or more layers. The network consists of 50 layers, including convolutional layers, batch normalization layers, and ReLU activation functions. It also uses a bottleneck design with three layers in each residual block: 1x1, 3x3, and 1x1 convolutions. ResNet50 is known for its high accuracy and efficiency. It has been successfully used in various image recognition tasks and competitions, and it won first place at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015. The main characteristics of the ResNet50 model are:

- **Residual Blocks:** The core idea is the use of residual blocks, where the output of a few stacked layers is added to the input of those layers. This helps in addressing the degradation problem in deep networks.
- **Identity Shortcuts:** These shortcuts perform identity mapping and are added to the outputs of the stacked layers. They introduce neither extra parameters nor computational complexity.

- **Bottleneck Design:** For deeper networks, a bottleneck design is used, which involves a stack of three layers (1x1, 3x3, 1x1 convolutions) to reduce and then restore dimensions, making the network more efficient.
- **Network Depth:** The architecture includes very deep networks, such as 50, 101, and 152 layers, which are significantly deeper than previous models like VGG16, yet more efficient in terms of computational complexity.

More details about the network's architecture are listed below, while a relevant diagram is provided in Figure 14.

- **Input Layer:** The input to ResNet50 is an image of size 224x224x3.
- **Convolutional Layers:** The network starts with a 7x7 convolutional layer with 64 filters and a stride of 2, followed by a 3x3 max pooling layer. This is followed by a series of residual blocks, each containing three layers: 1x1, 3x3, and 1x1 convolutions. The 50 in "ResNet50" stands for the 50 layers that the model has in total.
- **Fully Connected Layer:** After the convolutional layers, there is a global average pooling layer that reduces the spatial dimensions to 1x1. This is followed by a fully connected layer with 1000 neurons.
- **Output Layer:** The final layer is a softmax layer that outputs probabilities for 1000 classes, corresponding to the ImageNet dataset.

This architecture allows ResNet50 to achieve high accuracy while maintaining manageable computational complexity.

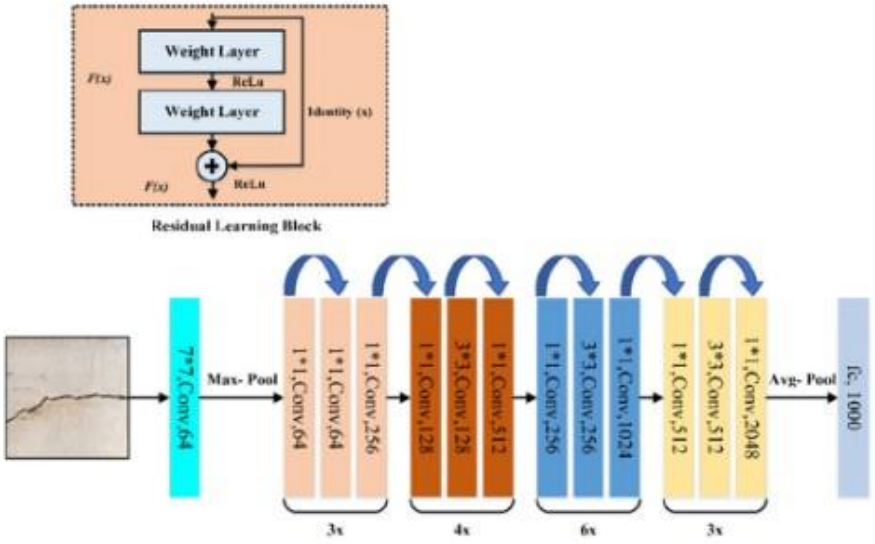


Figure 14. ResNet50 model architecture (Ali et al., 2021)**.**

2.3.3 Densely Connected Convolutional Network (DenseNet121)

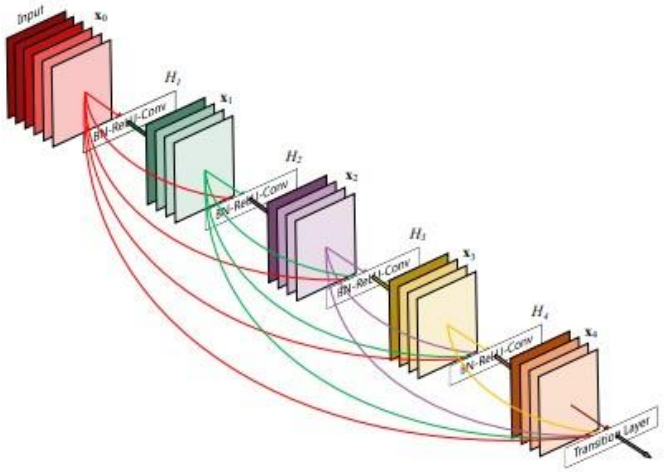
DenseNet121 is a type of Dense Convolutional Network (DenseNet) designed to improve the flow of information and gradients through the network, making it more efficient and easier to train. Its architecture was introduced by Huang et al. in 2016 (Huang et al., 2016). The key points of DenseNet121 model are:

- **Architecture:** It consists of 121 layers, including convolutional layers, dense blocks, and transition layers. Each dense block connects each layer to every other layer in a feed-forward fashion. Its architecture is analyzed further below.
- **Efficiency:** DenseNet121 is designed to be highly parameter-efficient, requiring fewer parameters than traditional convolutional networks while maintaining high performance.
- **Performance:** It achieves state-of-the-art results on various benchmark datasets like CIFAR-10, CIFAR-100, SVHN, and ImageNet.
- **Advantages:** DenseNets alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and reduce the number of parameters needed.

Below is a brief overview of its architecture.

- **Input Layer:** The input to DenseNet-121 is an image of size 224x224 pixels.
- **Convolutional Layer:** The initial convolution layer has 2k (64 due to the fact that growth rate k is equal to 32) filters of size 7x7 with a stride of 2, followed by a 3x3 max pooling layer with a stride of 2.
- **Dense Blocks:** There are four dense blocks, each consisting of multiple layers. Each layer within a dense block receives inputs from all preceding layers and passes its own feature-maps to all subsequent layers. As stated, the growth rate (k) is 32.
- **Transition Layers:** Between dense blocks, transition layers perform 1x1 convolutions followed by 2x2 average pooling to reduce the size of feature-maps.
- **Fully Connected Layer:** After the final dense block, a global average pooling layer is applied, followed by a fully connected layer with 1000 units and a softmax activation function for classification.

DenseNet121 architecture ensures efficient feature reuse and reduces the number of parameters compared to traditional convolutional networks (Huang et al., 2016).



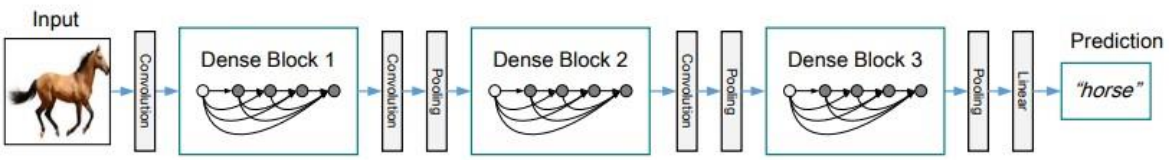


Figure 15. DenseNet architecture. Above figure depicts a 5 layer dense block with growth rate (k) 4. The second figure depicts a DenseNet with three dense blocks. The layers between two neighboring blocks are called transition layers and change feature-map sizes via convolution and pooling (Huang et al., 2016).

2.3.4 Inception Network

The Inception model, also known as GoogLeNet, is a deep convolutional neural network architecture designed to improve the utilization of computing resources within the network. It was introduced by researchers at Google and achieved state-of-the-art performance in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) (Szegedy et al.,2014).

Below is a brief overview of the model's architecture:

- **Input Layer:** The input to the network is an image of size 224x224 with RGB color channels.
- **Convolutional Layers:** The network starts with a 7x7 convolutional layer followed by max-pooling. It includes multiple Inception modules, each consisting of 1x1, 3x3, and 5x5 convolutions, along with max-pooling layers. These modules are designed to capture features at different scales.
- **Fully Connected Layers:** Instead of traditional fully connected layers, the network uses average pooling followed by a linear layer. This reduces the number of parameters and helps in better generalization.
- **Output Layer:** The final layer is a softmax classifier that outputs probabilities for 1000 classes.

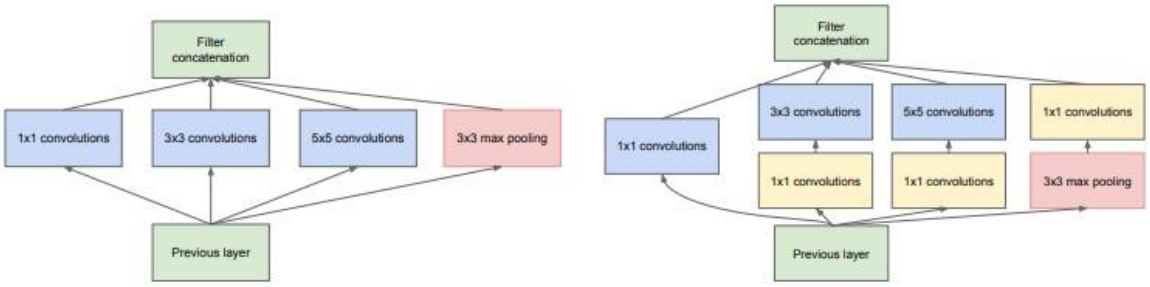


Figure 16. Inception architecture. Left figure (16a): Inception module, naive version. This version combines multiple convolutional layers (1x1, 3x3, 5x5) and a 3x3 max pooling layer. The outputs of these layers are concatenated to form the input for the next stage. This approach can lead to a large number of outputs, increasing computational complexity. **Right figure (16b):** Inception module with dimensionality reduction. This version introduces 1x1 convolutions before the 3x3 and 5x5 convolutions to reduce the number of input channels. This reduces computational cost while maintaining performance, making the network more efficient. These modules help the network handle multiple scales of information efficiently and effectively (Szegedy et al., 2014).

The Inception model's design allows for increasing the depth and width of the network while keeping the computational budget constant, making it efficient and powerful for image classification and detection tasks. More specifically, the advantages of using the Inception model are:

- **Efficient use of resources:** The architecture allows for increasing the depth and width of the network while keeping the computational budget constant, making it suitable for real-world applications.
- **Improved accuracy:** By combining deep architectures with classical computer vision techniques, the Inception model achieves higher accuracy in image classification and object detection.
- **Scalability:** The use of dimension reduction techniques, such as 1x1 convolutions, helps manage computational complexity, allowing the network to scale effectively.
- **Versatility:** The model's design supports various scales of visual information processing, making it adaptable to different tasks and datasets (Szegedy et al., 2014).

3 Chapter 3: Relative Work

This chapter reviews the existing research in the field of fracture detection using machine learning and deep learning techniques, with a focus on two specific types of fractures: wrist fractures and femoral intertrochanteric fractures. By examining previous studies, this chapter highlights the advancements made in automated diagnostic systems and their application in medical imaging. The first section discusses wrist fracture detection, showcasing an approach that has been used to accurately identify fractures from radiographic images. The second section discusses femoral intertrochanteric fracture detection, emphasizing the challenges and successes in applying artificial intelligence to this more complex fracture type.

3.1 Wrist Fracture Detection

Robert Lindsey et al. proposed a deep learning model for detecting fractures and localizing them, based on radiographs (Lindsey et al., 2018). For the purpose of developing a model, a collection of radiographs was obtained retrospectively from a specialty hospital in the United States. Orthopedic surgeons provided clinical interpretations for these radiographs using a tool to draw bounding boxes around fractures. A deep learning model was designed to detect and localize fractures in radiographs and was trained based on the labels accompanying the dataset. The model's performance was then clinically tested on two separate datasets. To evaluate whether the model can assist emergency medicine clinicians in fracture detection, a controlled experiment was conducted.

The radiographs used in the study were obtained from the Hospital for Special Surgery (HSS) between September 2000 and March 2016. The dataset included 135,845 radiographs of various body parts, with 34,990 of them being wrist radiographs. Two test datasets were used for evaluating the model's performance. The ground truth labels for fracture presence and location were assigned by orthopedic surgeons. The model development involved a two-stage training process: a bootstrapping stage using a large dataset of radiographs from various body parts and a specialization stage using wrist radiographs. In total, 132,345 radiographs were used for training the model.

The model used a deep convolutional neural network for fracture detection and localization. It employed a dual output approach, with one output providing a binary classification and the other output generating a heat map indicating the probability of fractures at specific locations in the radiographs. The model's performance was evaluated using receiver operating characteristic (ROC) curves and the area under the curve (AUC) on test datasets.

Clinicians were also assessed for their diagnostic accuracy with and without the model's assistance in a controlled experiment. The study reports the model's diagnostic performance, the sensitivity and specificity of clinicians, and the time it took clinicians to read radiographs in the experiment.

3.1.1 Results

The results of the study show that the trained model demonstrated excellent performance in detecting and localizing fractures in wrist radiographs. On Test Set 1, the model achieved an AUC of 0.967, and on Test Set 2, it achieved an AUC of 0.975. In a subset of images in Test Set 2 where there was no uncertainty about the reference standard, the model achieved an impressive AUC of 0.994. This indicates a high level of agreement between the model's assessments and the reference standard provided by senior subspecialized orthopedic hand surgeons. The model's ability to precisely identify the presence and location of visible fractures is also noted.

The study further evaluated the impact of the deep learning model on the diagnostic accuracy of emergency medicine clinicians. Both emergency medicine medical doctors (MDs) and physician assistants (PAs) showed significant improvements in sensitivity and specificity when aided by the model. The average sensitivities and specificities for these clinicians were substantially enhanced when using the model compared to unaided interpretations. The average reduction in misinterpretation rate across clinicians was 47.0%. Almost every clinician exhibited improvements in both sensitivity and specificity.

Additionally, the model's performance was compared to that of the clinicians. On the same images, the model operated at 93.9% sensitivity and 94.5% specificity under its predetermined decision threshold and achieved an AUC of 0.990. This suggests that the model's performance was competitive with or better than that of the clinicians.

When assessing the clinicians' diagnostic accuracy, it was found that, without the model's assistance, the average sensitivity for emergency medicine MDs was 82.7%, and the unaided specificity was 87.4%. Also, for emergency medicine PAs, the unaided sensitivity was 78.0%, and the unaided specificity for PAs was 87.5%.

When the clinicians were aided by the deep learning model, their diagnostic sensitivities significantly improved. For emergency medicine MDs, the model-enhanced sensitivity was 92.5% and specificity 94.1%, representing a notable increase in fracture detection accuracy. Similarly, for emergency medicine PAs, the model-enhanced sensitivity was 89.9% and specificity 93.6%, demonstrating a substantial improvement in their diagnostic performance.

The study also investigated the relationship between reading time and diagnostic accuracy. It was observed that radiographs that were read quickly without assistance were generally interpreted accurately. However, as the reading time increased, the diagnostic accuracy for both aided and unaided conditions deteriorated. Notably, the difference in accuracy between the aided and unaided reading conditions increased with longer unaided reading times, indicating that emergency medicine workers dealing with challenging and time-consuming cases would benefit more from the computer-aided detection (CAD) software.

These results provide strong evidence for the effectiveness of the deep learning model in assisting clinicians in fracture detection and localization, ultimately improving diagnostic accuracy, especially in challenging cases and time-sensitive situations.

3.2 Femoral Intertrochanteric Fracture Detection

In the study conducted by Liu et al. (Liu et al., 2022), x-ray data for femoral intertrochanteric fractures (FIF) were collected from five hospitals. The dataset comprised 700 x-rays from 459 FIF patients, including both fractured (459 images) and normal hips (241 images). To ensure an accurate and unbiased model, the dataset was split using a 9:1 ratio into a training set of 643 images and a test set of 57 images. Additionally, physicians manually labeled the images to mark fracture lines.

For the development of the diagnostic algorithm, a Faster-RCNN target detection model was employed. The model underwent data augmentation processes, including image rollover, rotation, cropping, and blurring, which expanded the dataset from 643 to 3,215 images. The dataset was then used to train the algorithm, which focused on learning both the anatomical structure of normal hips and the features specific to fracture lines in FIF x-rays. The model's architecture included a convolutional neural network (CNN) for feature extraction and a Region Proposal Network (RPN) to localize potential fractures. By pooling and refining the image regions, the algorithm was able to output predictions, classifying x-rays as either fractured or normal. The performance of the model was evaluated using key metrics such as accuracy, sensitivity, specificity and misdiagnosis rate, with the results compared against a panel of five orthopedic attending physicians.

This comparison revealed how the algorithm could potentially assist in diagnostic settings. Despite the expertise of the physicians involved, the Faster-RCNN algorithm demonstrated competitive performance in identifying fractures, offering an accurate and efficient tool for clinical practice. This work highlights the clinical feasibility of implementing artificial intelligence into diagnostic workflows, indicating that such systems could support medical professionals by reducing diagnostic time and improving the consistency of fracture detection.

3.2.1 Results

After the algorithm was trained using the expanded dataset, it was tested on the separate test data to evaluate its performance in detecting FIFs. For images identified as containing fractures, the algorithm highlighted the suspicious fracture lines with a red rectangle. Various performance metrics were used to assess the model’s effectiveness, including the F1 score, recall, precision, average precision (AP), mean average precision (mAP), intersection over union (IoU), area under the curve (AUC), and receiver operating characteristic (ROC) curve.

The final results of the algorithm demonstrated a strong performance in classifying FIF and normal hips. The accuracy of the model was 0.88. The sensitivity, or the model’s ability to correctly identify actual fracture cases, was 0.89, while the missed diagnosis rate was 0.11, meaning that the algorithm failed to detect 11% of fracture

cases. The specificity, which measures the ability to correctly identify normal hips, was 0.87, and the misdiagnosis rate was 0.13. These results suggest that the algorithm is capable of providing reliable diagnostic support, with a low rate of missed diagnoses and a reasonably high specificity, although some misdiagnoses were still present.

4 Chapter 4: Data

In this section, the data used for developing the proposed approach are being presented. Due to the fact that there was a limited number of open-source hip x-ray images that could be used for the models' training, it was also necessary to gather a dataset from various sources. The process of acquiring the dataset is described below.

4.1 Data Gathering

Since the aim of the thesis is to develop an approach that could effectively classify twelve classes of hip x-rays, a robust dataset consisting of data addressing all twelve classes needed to be gathered. Among many types of hip fractures, disorders and hip surgeries, after consulting orthopedic surgeon Zachariadis Christos, the following classes were considered to provide the most value (see Figure 17):

1. Left hip normal
2. Right hip normal
3. Left hip intertrochanteric fracture
4. Right hip intertrochanteric fracture
5. Left hip subcapital fracture
6. Right hip subcapital fracture
7. Left hip osteoarthritis
8. Right hip osteoarthritis
9. Left hip arthroplasty
10. Right hip arthroplasty
11. Left hip nailing
12. Right hip nailing

Normal left

Normal right

Intertrochanteric left

Intertrochanteric right

Subcapital left

Subcapital right


Arthroplasty left


Arthroplasty right


Nailing left


Nailing right


Osteoarthritis left


Osteoarthritis right

Figure 17. X-ray images from the dataset. Each image corresponds to one class.

Data was gathered from various sources on the web and from x-ray images provided by doctor Zachariadis along with the respective diagnoses. The images were carefully picked and annotated with his help and guidance, ensuring that no mistakes were made when annotating them. The following sources were used for getting the medical data:

- https://www.kaggle.com/datasets/ibombonato/xray-body-images-in-png-unifesp-competion
- https://radiopaedia.org/articles/hip-hemiarthroplasty?lang=us
- https://radiopaedia.org/articles/total-hip-arthroplasty?lang=us
- https://radiopaedia.org/articles/garden-classification-of-hip-fractures?lang=us
- https://radiopaedia.org/articles/vancouver-classification-of-periprosthetic-hip-fractures ?lang=us
- https://radiopaedia.org/articles/periprosthetic-fracture?lang=us
- https://radiopaedia.org/articles/osteoarthritis-of-the-hip?lang=us
- https://radiopaedia.org/cases/normal-hip-x-rays
- https://radiopaedia.org/cases/normal-pelvic-radiograph-female
- https://boneandspine.com/intertrochanteric-fractures/
- https://boneandspine.com/hip-injuries-xrays-and-photographs/
- http://www.boneschool.com/hip/hip-fractures/intertrochanteric-fractures
- https://www.sciencedirect.com/science/article/pii/S1063458406003281
- https://www.nature.com/articles/s41598-020-70660-4
- https://www.researchgate.net/publication/51434368_Evaluation_of_Bernese_periacetabular_osteotomy_Prospective_studies_examining_projected_load-bearing_area_bone_density_cartilage_thickness_and_migration
- https://www.futuremedicine.com/doi/10.2217/fmeb2013.13.198

4.2 Data preprocessing

Since all data are medical images, not much preprocessing was required in order for the model to be trained. The reason is that the data consist of medical images and the features should not be distorted much. In order to achieve that, images were resized to shape 256x256. Then, the pixel values of each image were normalized to the range [0,1]. Additionally, the labels were assigned to each image and the processed images were returned as 3-dimensional numpy arrays with shape [256,256,3]. After the preprocessing, the total count of the data was

812 arrays. Finally, the dataset was split into five subsets of data, one for each classification step, which will be analyzed in the following chapter.

4.3 Data compliance with GDPR

The General Data Protection Regulation (GDPR) (*General Data Protection Regulation (GDPR) Compliance Guidelines*, n.d.) plays a critical role in ensuring the security and privacy of medical data. Given the sensitivity of health-related information, strict regulations are required to protect patient confidentiality and prevent misuse. The GDPR establishes clear guidelines for handling personal data, mandating that it should be anonymized or pseudonymized. In the context of medical research, compliance with GDPR fosters trust between patients and institutions by ensuring that patient data is handled ethically and securely. This regulation helps prevent unauthorized access to sensitive health information, thus maintaining the integrity of both clinical and academic research.

In this study, all images that were provided by doctor Zachariadis were anonymized. The original x-rays were photographed with the use of a mobile phone device, and then the resulting photographs were provided for the dataset, instead of the original x-rays. Thus, the final images have a completely different structure than that of the DICOM images, and the metadata do not correlate with any information about the patients. The final folder that was provided with the aforementioned data contains images of png, .jpeg and .jpg format. Therefore, the complete anonymization of the dataset was ensured with no possibility of tracing the images back to the patients, entirely complying with the GDPR. To conclude, the data that were incorporated in the final dataset, originated from all sources, are uniform, in .png, .jpg and .jpeg formats and completely anonymized (*What Is Personal Data?*, n.d.).

5 Chapter 5: Methodology & Experiments

The classification process is split into five classification stages (Figure 18). Different models are being used for each classification stage with different architectures, depending on the complexity of the images and the difficulty of the model to find meaningful features in order to classify each subset of data. The classification stages of the pipeline are:

- **First classification stage** is a binary classification. The first set of data is used during this phase, which consists of all images. The model classifies if the x-ray image contains a left hip or a right hip.
- **Second classification stage** is a binary classification and the second set of data is used. All data are also contained in this set, normal images are contained in the first class and the rest of the images in the second class (e.g. arthroplasty, intertrochanteric fracture). This stage classifies whether the image contains a normal (healthy) hip or a not normal hip (operated or not operated). If the classification output is "normal", the process does not move on to the next stages.
- **Third classification stage** is a binary classification. The third set of data, which is used for training the models in this stage, is a subset of the entire dataset. The first class includes x-rays of operated hips, containing arthroplasty and nailing images, while the second class includes images of not operated hips. Not operated hips include cases of subcapital fractures, intertrochanteric fractures and osteoarthritis.
- **Fourth classification stage** is a binary classification and is initiated only if the output of the third classification stage is "operated". The fourth subset of data is used, which encompasses images of arthroplasty and nailing.
- **Fifth classification stage** is a multi-class classification and is initiated only if the output of the third classification stage is "not operated". For this stage, the fifth subset of data is used for training, which consist of subcapital fractures, intertrochanteric fractures and osteoarthritis x-ray images.

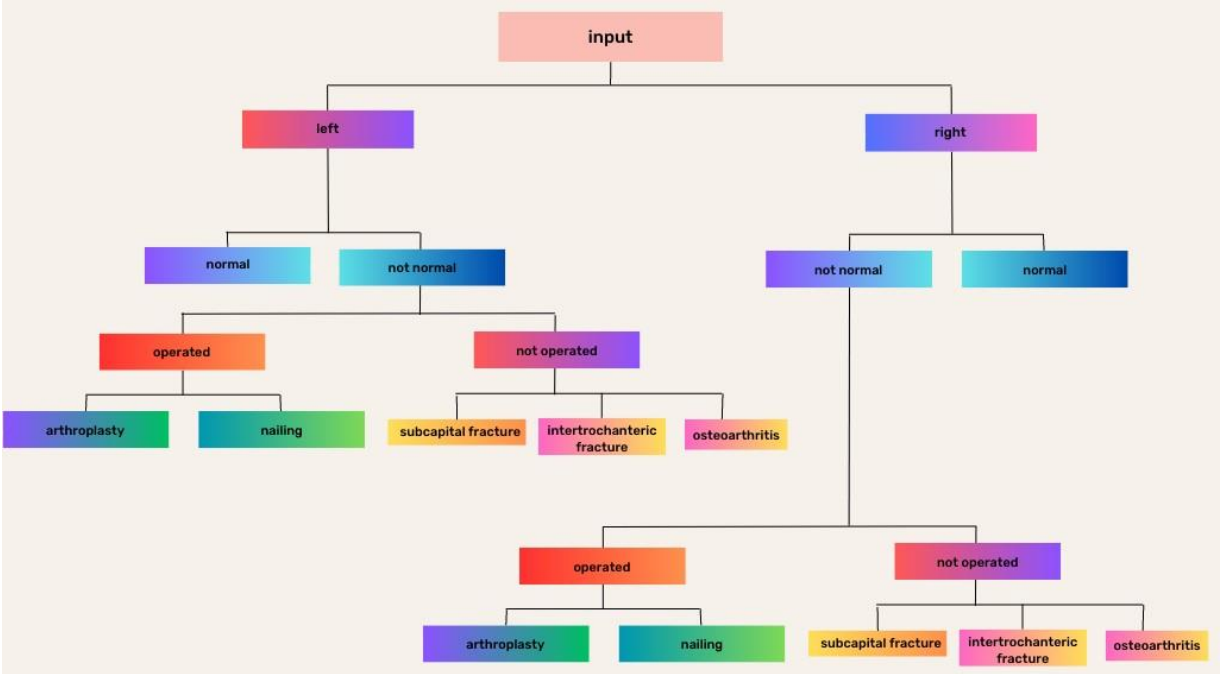


Figure 18. Classification pipeline. Each level corresponds to one classification step, where the classification is conducted from different models.

5.1 Left / Right Hip Classification Stage

This is the first stage of the classification pipeline. Various models were trained and tested, with the ResNet50 model demonstrating the best accuracy.

5.1.1 DenseNet Model

5.1.1.1 Data preparation, Architecture and Training

First, data is split in training, validation and test sets. In the beginning, the original dataset is split into training and validation sets using the "train_test_split" method that the scikit-learn library offers. 70% of the 812 images were used as the training data (568 images). The split is stratified, meaning the distribution of classes in the train and validation sets is the same as in the original dataset. Then, the rest 30% of the data is further split into validation and test sets. From the remaining 244 images, 65% of the images formed the validation set (158 images) and 35% the test set (86 images).

In this implementation, the DenseNet121 model is used with its pre-trained weights from the ImageNet dataset, which allows the model to generalize features easier. By setting the parameter *"include_top=False"*, the fully connected layers at the top of the DenseNet121 base model are excluded, leaving only the convolutional layers. Additionally, the DenseNet121 layers are frozen, meaning that during training, their weights will not be updated.

Additionally to the DenseNet121 main architecture, the model architecture also includes several custom layers, which will process the feature maps generated by DenseNet121. The input layer accepts 256x256 pixel images with three color channels. The next layer is a Global Average Pooling layer, which reduces each feature map to a single value by computing the average. This step reduces the dimensionality of the feature maps, converting them into a 1D vector. Later on, a Batch Normalization layer is placed to normalize the output from the previous layer. After that, a Flatten layer follows, preparing the data for the fully connected layers. The first two Dense layers each have 32 neurons and use the ReLU activation function. These layers are followed by a Dense layer with 16 neurons and another with 8 neurons. The output layer is a Dense layer with two neurons, using a sigmoid activation function, since it is a binary classification task (Figure 19).

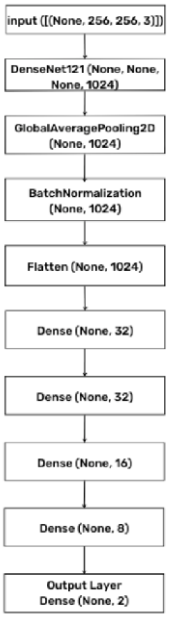


Figure 19. DenseNet model architecture.

A critical component of this training process is the use of early stopping. Early stopping is a regularization technique that monitors the model's performance on the validation dataset during training, in this case the validation loss. Purpose of this technique is to stop the training process when the model stops improving, which typically prevents overfitting. Additionally, the training will stop if the validation loss does not improve for five consecutive epochs. Also, the early stopping is configured to start monitoring only after the first 10 epochs. This allows the model to stabilize and begin learning meaningful patterns before the early stopping mechanism begins to monitor its performance.

The Adam optimizer is chosen for training. After trying different learning rates, the learning rate was set to 0.001. Also, the loss function used is binary cross-entropy, which is suitable for binary classification tasks. Additionally, the training is done with a batch size of 16 and the model is being trained for a maximum of 100 epochs. However, due to the early stopping mechanism, the actual number of epochs is fewer. The validation loss had stopped improving and the training terminated at epoch 18. The “shuffle” parameter was also set to “True”, which means that the training data is shuffled before each epoch.

5.1.1.2 Results

When the model stopped training due to the early stopping mechanism, the training results shown in Table 3, indicate that the model was prone to overfitting. This can be also seen in Figure 20. Several changes were made to the model trying to achieve better results, but did not yield better results.

Data subset	**Accuracy**	**Loss**
Training	0.9818	0.0437

Validation	0.7215	1.3570

Table 3. DenseNet121 model training results.

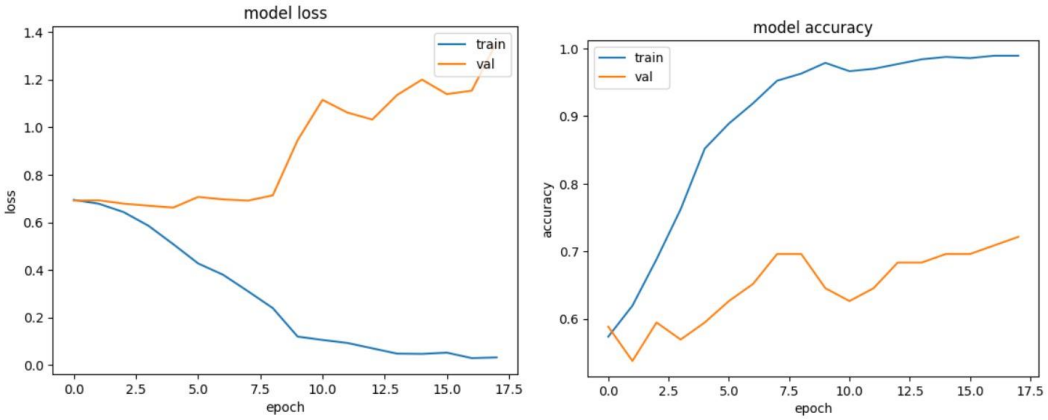


Figure 20. DenseNet training loss and accuracy. DenseNet121 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

Finally, the accuracy of the model on the test set was 0.72. The confusion matrix in Figure 21 and the classification report in Table 4 shows the model's results in more detail.

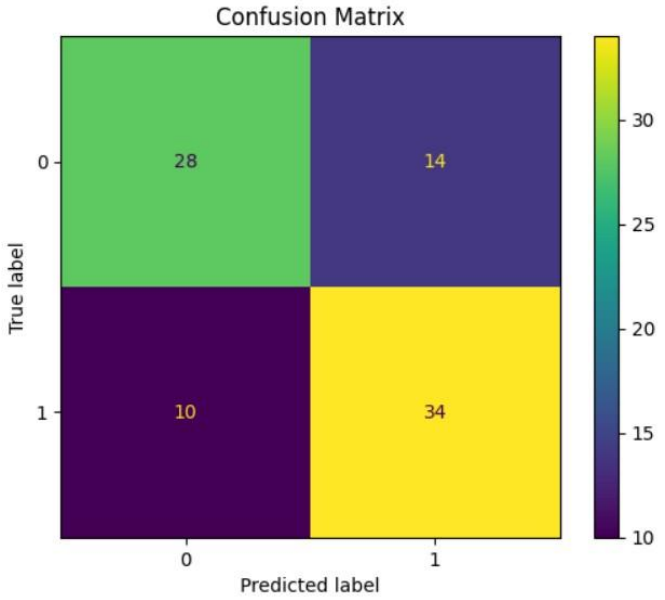


Figure 21. Classification report and confusion matrix of the DenseNet121 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.74	0.67	0.70	42
1	0.71	0.77	0.74	44

accuracy			0.72	86
macro avg	0.72	0.72	0.72	86
weighted avg	0.72	0.72	0.72	86

Table 4. DenseNet121 classification report.

5.1.2 Inception Model

5.1.2.1 Data preparation, Architecture and Training

The dataset is split into training and validation sets using "train_test_split". 70% of the 812 images were split into training data (568 images). The split is once again stratified, keeping the distribution of classes in the train and validation, in accordance to their distribution the original dataset. Out of the 244 images in the validation set, 65% of the images remained in the validation set (158 images) and 35% were splitted to be used as the test set (86 images).

In this experiment, the Inception model is also used with its pre-trained weights from the ImageNet dataset. Additionally, the parameter *"include_top"* is set to *"False"* and the model's layers are frozen, leaving the weights not to be updated.

Below the Inception architecture, several custom layers are also included. The input layer accepts 256x256 pixel images with three color channels. The following layers include a Global Average Pooling layer. Following, a Batch Normalization layer is placed to normalize the output from the previous layer and a Flatten layer, preparing it for the fully connected layers. Later, two Dense layers are added, having 32 neurons each and using the ReLU activation function. Then, a Dense layer with 16 neurons and another with 8 neurons are added, both using the ReLU activation function. Finally, the output layer is a Dense layer with two neurons, using the sigmoid activation function (Figure 22).

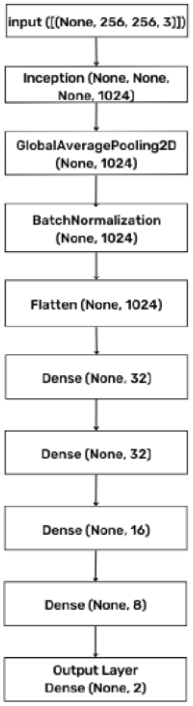


Figure 22. Inception model architecture.

In this training process, the Early Stopping technique is also utilized. Specifically, it monitors the model's validation loss during training. The training will stop if the validation loss does not improve for five consecutive epochs. Once again, the early stopping is set to start the monitoring after the 10th epoch.

The Adam optimizer is also chosen for training. The learning rate was set to 0.001 and the loss function used is binary cross-entropy. Also, the training is done with a batch size of 16 and the model is going to train for a maximum of 100 epochs. However, due to the early stopping mechanism, the actual number of epochs is fewer. More specifically, the validation loss stopped improving at epoch 16. Finally, the "shuffle" parameter was also set to "True", to shuffle the training data before each epoch.

5.1.2.2 Results

When the model stopped training due to the early stopping mechanism, the training results shown in Table 5, indicate that the Inception model also overfitted. This can be also seen in Figure 23. Several changes were made to the model trying to achieve better results, but these were the best along this experiment.

Data subset	**Accuracy**	**Loss**
Training	0.9298	0.1802
Validation	0.6076	1.2729

Table 5. Inception model training results.

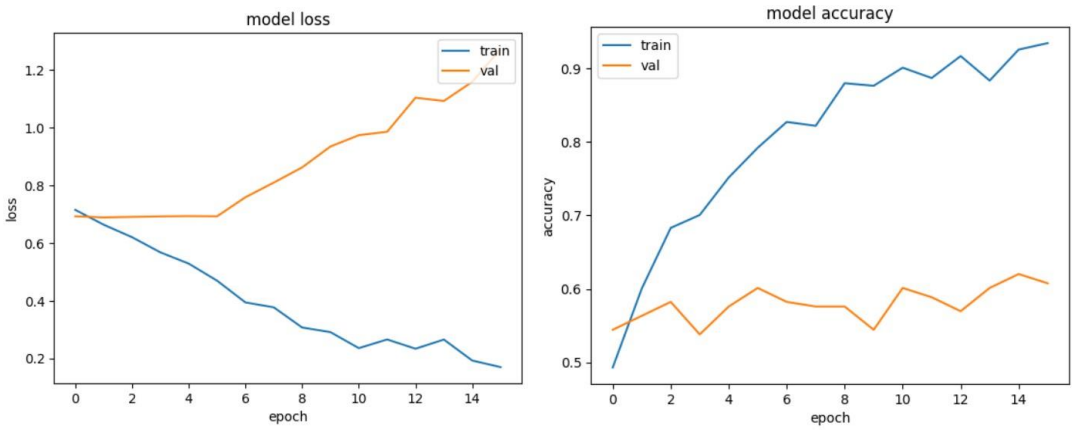


Figure 23. Inception training loss and accuracy. Inception model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

Finally, the accuracy of the model on the test set was 0.66. The confusion matrix in Figure 24 and the classification report in Table 6 shows the model's results in more detail.

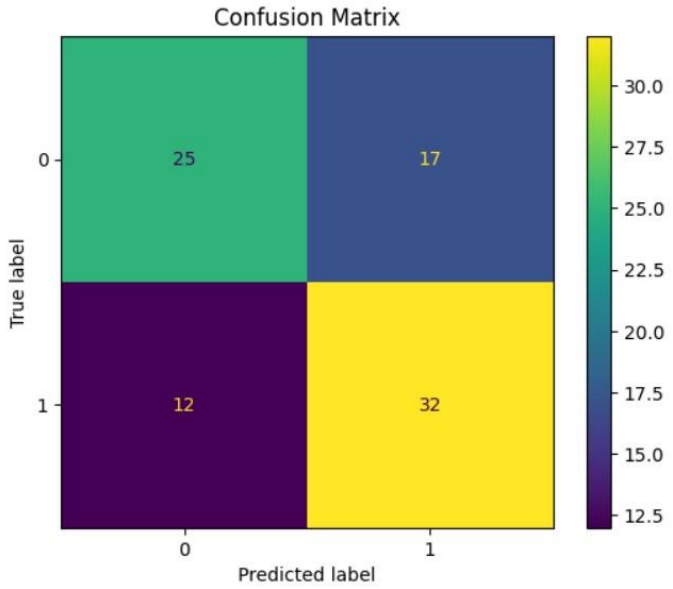


Figure 24. Confusion matrix of the Inception model's predictions on the test set.

	precision	recall	f1-score	support
0	0.68	0.60	0.63	42
1	0.65	0.73	0.69	44
accuracy			0.66	86
macro avg	0.66	0.66	0.66	86

weighted avg	0.66	0.66	0.66	86

Table 6. Inception classification report.

5.1.3 ResNet50 Model

The ResNet50 model had the best results amongst all the models that were implemented for the Left/Right classification task.

5.1.3.1 Data preparation, Architecture and Training

First of all, the dataset is split into training and validation sets. 70% of the 812 images were split into training data (568 images). Then, the validation set is further split into validation and test sets. From the 244 images in the validation set, 65% of the images remained in the validation set (158 images) and 35% in the test set (86 images). The split in this experiment is also stratified.

The ResNet50 model is also used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to False and the model's layers are frozen.

After the ResNet50 architecture, several custom layers are also added. The input layer accepts 256x256 pixel images with three color channels. The following layers include a Global Average Pooling layer. Following, a Batch Normalization layer is placed to normalize the output from the previous layer and then a Flatten layer. Later, two Dense layers are added, each one having 32 neurons and using the ReLU activation function. Then, a Dense layer with 16 neurons and another with 8 neurons are added, both using the ReLU activation function. Finally, the output layer is a Dense layer with two neurons, using a sigmoid activation function (Figure 25).

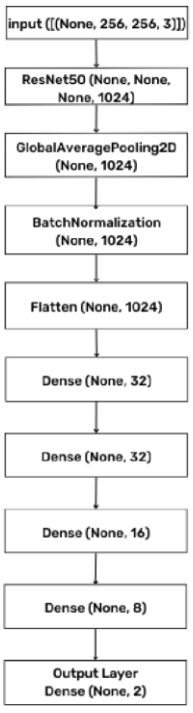


Figure 25. ResNet50 model architecture.

The Early Stopping technique is also utilized, which also monitors the model's validation loss during training. The training will stop if the validation loss does not improve for five consecutive epochs. Additionally, the early stopping is set to start the monitoring after 10 epochs.

The Adam optimizer is also chosen for training. The learning rate was set to 0.001 and the loss function used is binary cross-entropy. Also, the training is done with a batch size of 16 and the model is going to train for a maximum of 100 epochs. However, due to the early stopping mechanism, the validation loss stopped improving at epoch 20. Finally, the "shuffle" parameter was also set to "True".

5.1.3.2 Results

When the model stopped training due to the early stopping mechanism, the training results shown in Table 7, indicate that the ResNet50 model did not overfit, but had some minor fluctuations. This can be also seen in Figure 26. In general, the ResNet50 model had reliable results, making it the best model that was trained among the three models that were trained for this classification task. Finally, the accuracy of the model on the test set was 0.89. The confusion matrix in Figure 27 and the classification report in Table 8 shows the model's results in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.8896	0.2331
Validation	0.8734	0.4326

Table 7. ResNet50 model training results.

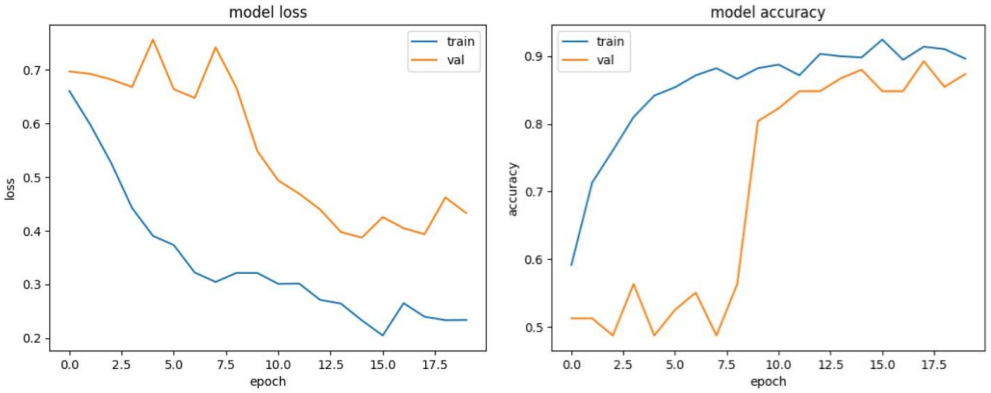


Figure 26. ResNet50 training loss and accuracy. ReNet50 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

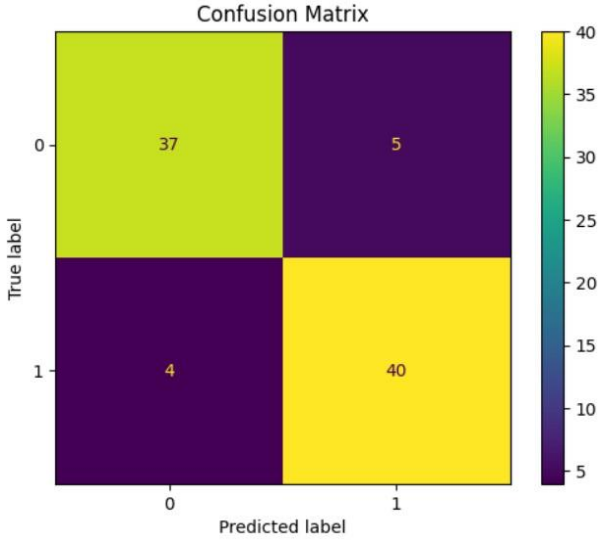


Figure 27. Confusion matrix of the ResNet50 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.90	0.88	0.89	42
1	0.89	0.91	0.90	44
accuracy			0.90	86
macro avg	0.90	0.90	0.90	86
weighted avg	0.90	0.90	0.90	86

Table 8. ResNet50 Classification report.

5.2 Normal / Not Normal Hip Classification Stage

This is the second stage of the classification pipeline. During this phase, it should be determined if the hip is normal or not normal. Various models were trained and tested for this task, but the VGG16 model achieved the best recall score.

5.2.1 Convolutional Neural Network

For the task of deciding whether a hip is normal or not, a custom made CNN architecture was trained and evaluated. In this subsection the proposed custom CNN will be presented.

5.2.1.1 Data preparation, Architecture and Training

First and foremost, the dataset is split into training and validation sets. 568 images (70% of the 812 images in the dataset) were used as the training set. The split is also stratified. After splitting to train (70%) and validation set (30%), the second set is further split into validation and test sets. From the 244 images in the

validation set, 65% of the images remained in the validation set (158 images) and 35% were used as the test set (86 images).

Concerning the CNN's architecture, due to the fact that the classification task is not very complicated, the model is not very deep and a reasonable number of neurons are included. To start, the input layer accepts 256x256 pixel images with three color channels. It is a 2D convolution with 8 neurons, has a kernel size of (3,3) and stride equal to 1. In addition, the parameter “padding” is equal to “same” and ReLU is being used as an activation function. The following layers include a Batch normalization layer and a 2D Max Pooling layer, with a kernel size of (3,3), stride equal to 2, and padding equal to “valid”. Following, another 2D Convolution layer is placed with 8 neurons, kernel size of (3,3), stride equal to 1, padding equal to “same” and ReLU as an activation function. The following two layers also include a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (3,3), stride equal to 2 and padding equal to “valid”. Next, a Dropout layer is added, with the purpose of randomly deactivating a portion of input units during each training update. This means that certain neurons are dropped out, along with their associated connections, with a probability of 0.1. Later, another 2D Convolution is added with 16 neurons, a stride of 1, a kernel size of (3,3), padding equal to “same” and using ReLU as the activation function. Also here, the two layers that follow are a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (3,3), stride equal to 2 and padding equal to “valid”. Then, a Dropout layer follows with a dropout probability of 0.1. Followed by a 2D Convolution with 8 neurons, kernel size of (3,3), stride equal to 1, padding equal to “same” and ReLU activation function. The three layers that follow are also a Batch Normalization layer, a 2D Max Pooling layer, with a kernel size of (3,3), stride equal to 2 and padding equal to “valid” and a Dropout layer with a dropout probability of 0.1. Before adding the fully connected layers, a Flatten layer is incorporated to reshape the data into a 1-dimensional array. With the use of this layer, the data have a valid shape to enter the fully connected layers. Therefore, four Dense layers are then added, with 32, 16, 16 and 8 neurons respectively, all using ReLU as the activation function. Finally, the output layer is a Dense layer with two neurons, using a sigmoid activation function. A simpler representation of this architecture can be seen in Table 9.

Layer	**Output Shape**
Conv2D	(None, 256, 256, 8)
BatchNormalization	(None, 256, 256, 8)
MaxPooling2D	(None, 127, 127, 8)
Conv2D	(None, 127, 127, 8)

BatchNormalization	(None, 127, 127, 8)
MaxPooling2D	(None, 63, 63, 8)
Dropout	(None, 63, 63, 8)
Conv2D	(None, 63, 63, 16)
BatchNormalization	(None, 63, 63, 16)
MaxPooling2D	(None, 31, 31, 16)
Dropout	(None, 31, 31, 16)
Conv2D	(None, 31, 31, 8)
BatchNormalization	(None, 31, 31, 8)
MaxPooling2D	(None, 15, 15, 8)
Dropout	(None, 15, 15, 8)
Flatten	(None, 1800)
Dense	(None, 32)
Dense	(None, 16)
Dense	(None, 16)
Dense	(None, 8)
Dense	(None, 2)

Table 9. Custom CNN architecture.

In the training process, Early Stopping is utilized, as it monitors the model's validation loss during training. The training will stop if the validation loss does not improve for five consecutive epochs. Additionally, the early stopping is set to start monitoring after the 10th epoch has been completed.

The Adam optimizer is chosen for training. The learning rate is set to 0.0001 and the loss function used is binary cross-entropy. Also, the training is done with a batch size of 16 and the model is going to train for a maximum of 100 epochs. However, due to the early stopping mechanism, as the validation loss stopped improving, the model

stopped training at the 38th epoch. Finally, the "shuffle" parameter was also set to "True".

5.2.1.2 Results

When the model stopped training, the training results shown in Table 10, indicate that the CNN did not overfit, and showcased satisfactory performance. The training loss and accuracy can be seen in Figure 28. Finally, the accuracy of the model on the test set was 0.95 and recall 0.94. The confusion matrix in Figure 29 and the classification report in Table 11 shows the model's predictions in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.9935	0.0328
Validation	0.9494	0.1563

Table 10. Custom CNN training results.

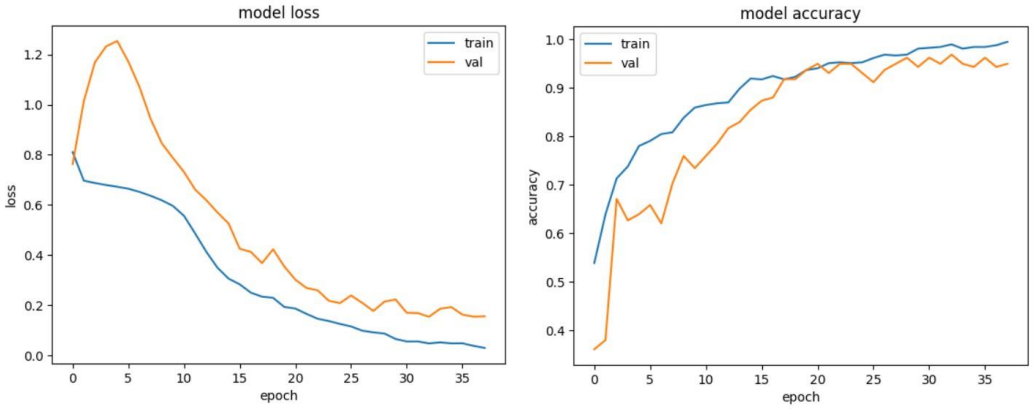


Figure 28. Custom CNN training loss and accuracy. Custom CNN model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

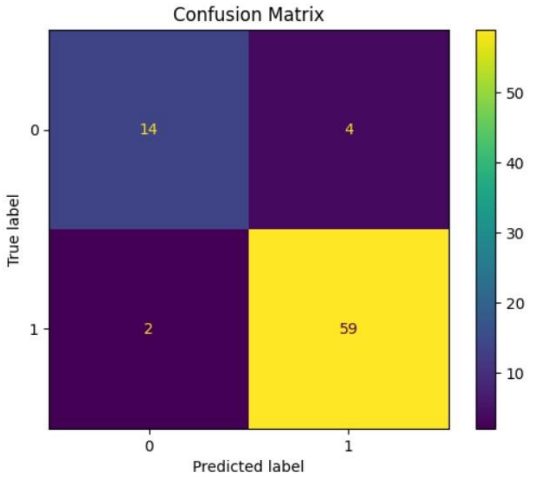


Figure 29. Confusion matrix of the CNN model's predictions on the test set.

	precision	recall	f1-score	support
0	0.91	0.91	0.91	22
1	0.97	0.97	0.97	64
accuracy			0.95	86
macro avg	0.94	0.94	0.94	86
weighted avg	0.95	0.95	0.95	86

Table 11. Custom CNN Classification report.

5.2.2 ResNet50 Model

5.2.2.1 Data preparation, Architecture and Training

The split of the training and test data is the same as established for training the CNN above (568 images for training, 158 for validation and 86 for testing). The ResNet50 model is used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false and the model's layers are frozen.

After the ResNet50 base architecture, several custom layers are also added. The input layer accepts 256x256 pixel images with three color channels. The following layer is a Global Average Pooling layer. Following, a Batch Normalization layer is placed to normalize the output from the previous layer, and then a Flatten layer. In continuation, four Dense layers are added with ReLU as the activation function, with each one having 32, 32, 16 and 8 neurons respectively. Finally, the output layer is a Dense layer with two neurons, using a sigmoid activation function (Figure 30).

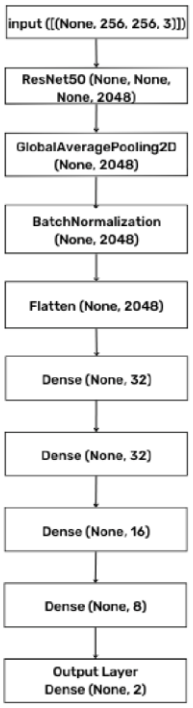


Figure 30. ResNet50 model architecture.

The Early Stopping technique is also utilized here, monitoring the model's validation loss during training. The training will stop if the validation loss does not improve for five consecutive epochs and the monitoring is set to start after 10 epochs.

The Adam optimizer is chosen for training. The learning rate was set to 0.001 and the loss function used is binary cross-entropy. Additionally, the training is done with a batch size of 16 and the model is going to train for a maximum of 100 epochs. Due to the early stopping, the training process halted when the loss stopped improving at epoch 26. Finally, the “shuffle” parameter was set to “True”.

5.2.2.2 Results

The results after the training process (shown in Table 12), indicate that the ResNet50 model had adequate results. This can be also seen in Figure 31, showing the model’s loss and accuracy across the training process. Although the model showed signs of overfitting during the last epochs, the accuracy of the model on the test set was better than the CNN’s, with an accuracy of 0.90 and recall 0.84. The confusion matrix in Figure 32 and the classification report in Table 13 shows the model's results in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.9812	0.0566
Validation	0.9304	0.3086

Table 12. ResNet50 model training results.

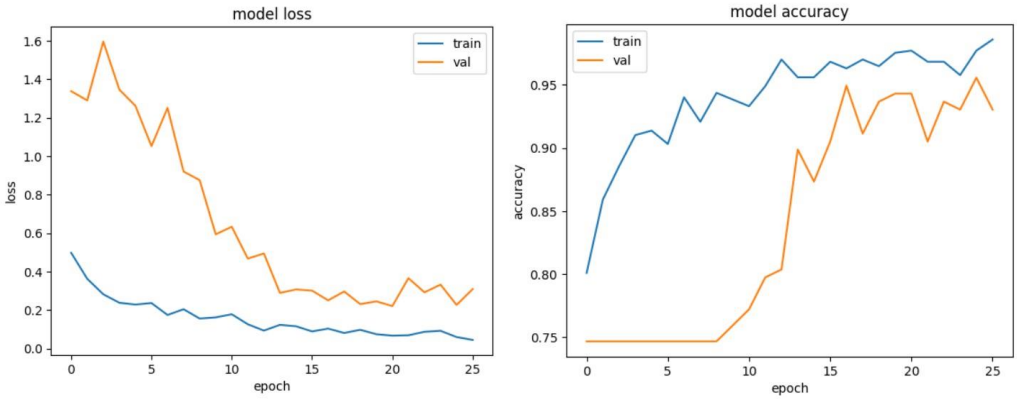


Figure 31. ResNet50 training loss and accuracy. ResNet50 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

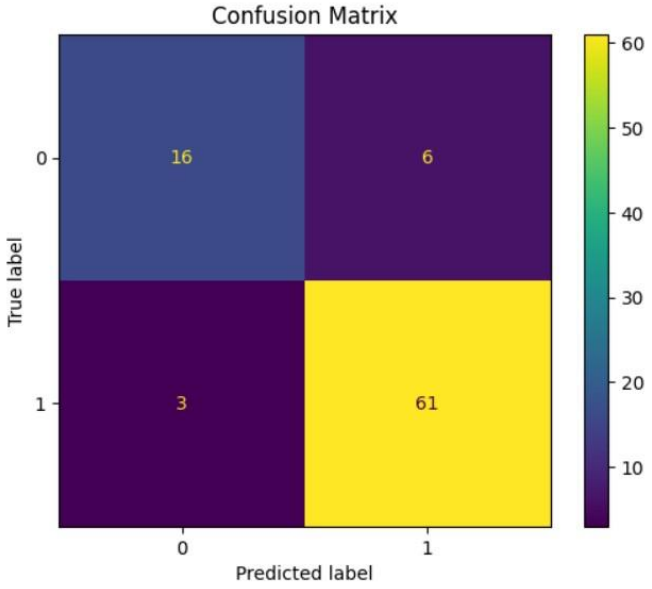


Figure 32. Confusion matrix of the ResNet50 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.84	0.73	0.78	22
1	0.91	0.95	0.93	64
accuracy			0.90	86
macro avg	0.88	0.84	0.86	86
weighted avg	0.89	0.90	0.89	86

Table 13. ResNet50 Classification report.

5.2.3 VGG16 Model

Among the three models that were trained for this classification stage, VGG16 had the best results.

5.2.3.1 Data preparation, Architecture and Training

The split of the training and test data is the same as the other two experiments for this classification stage, with 568 images for training, 158 for validation and 86 for testing, as mentioned above. The VGG16 model is also used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false and the model's layers are frozen.

After the VGG16 base model's architecture, several custom layers are added. The input layer receives inputs of 256x256 pixel images with three color channels. The following layers include a Global Average Pooling layer and a Batch Normalization layer. These layers are placed in order to reduce the dimensionality of the features and normalize the output from the previous layer, respectively. Then a Flatten layer follows. Later, four Dense layers are added with ReLU as the activation function, each one having 32, 32, 16 and 8 neurons in that exact order. Finally, the output layer is a Dense layer with two neurons, using a sigmoid activation function (Figure 33).

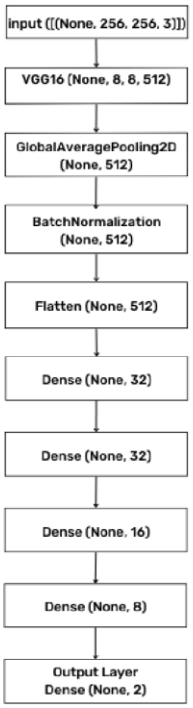


Figure 33. VGG16 model architecture.

Early Stopping is also utilized in this model, monitoring its validation loss during training. The training will also stop if the validation loss does not improve for five consecutive epochs and is set to start monitoring loss after 10 epochs.

The Adam optimizer is also chosen here for training. The learning rate is set to 0.001 and the loss function used is binary cross-entropy. During the training, the "shuffle" parameter is set to "True". Additionally, the training is conducted with a batch size of 16 and the model is going to train for a maximum of 100 epochs. Due to

the fact that the validation loss of this model stopped improving, Early Stopping stopped the training process at epoch 22.

5.2.3.2 Results

The training results after the training process ended (shown in Table 14), indicate that the model's training was stable, with minor fluctuations. This can also be seen in Figure 34, showing the model's loss and accuracy throughout the training process. The accuracy of the model on the test set was better than the other models that were implemented in this classification stage, with an accuracy of 0.99 and recall 0.98. The confusion matrix in Figure 35 and the classification report in Table 15 shows the model's results in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.9928	0.0177
Validation	0.9747	0.0787

Table 14. VGG16 model training results.

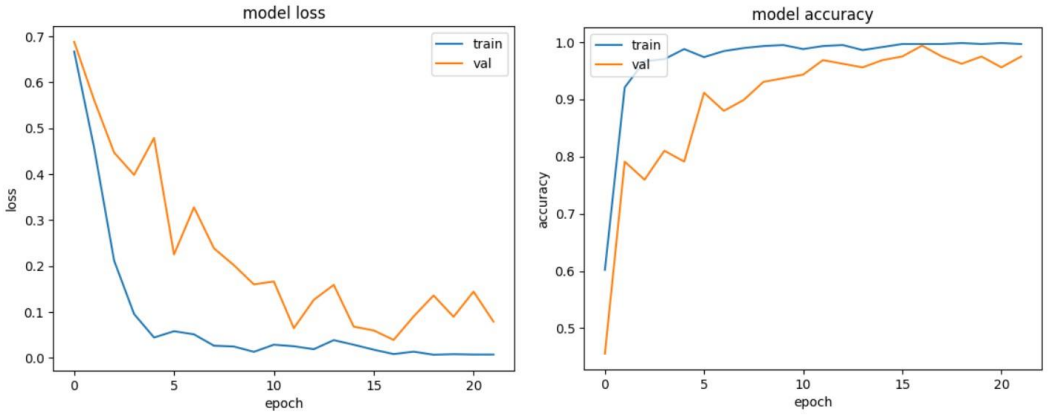


Figure 34. VGG16 training loss and accuracy. VGG16 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

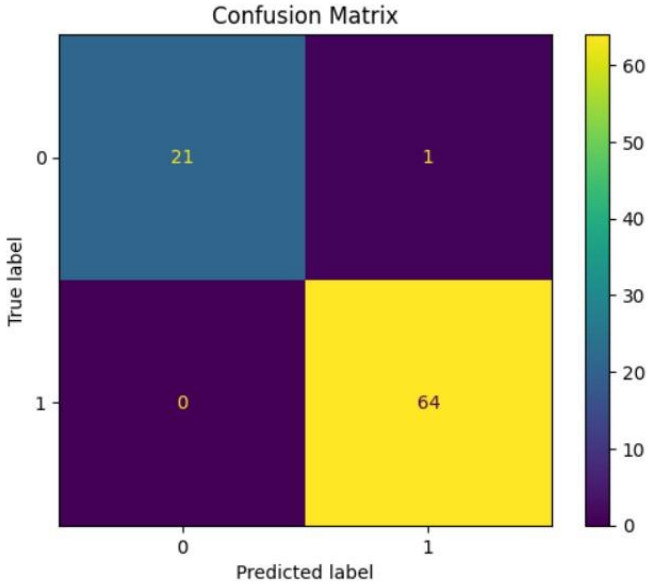


Figure 35. Confusion matrix of the VGG16 model's predictions on the test set.

	precision	recall	f1-score	support
0	1.00	0.95	0.98	22
1	0.98	1.00	0.99	64
accuracy			0.99	86
macro avg	0.99	0.98	0.98	86
weighted avg	0.99	0.99	0.99	86

Table 15. VGG16 Classification report.

5.3 Operated / Not Operated Hip Classification Stage

This is the third stage of the classification pipeline. The objective of this stage is to decide whether an x-ray of a not normal hip depicts an operated or a not operated hip. Various models were trained and tested for this specific task, but the VGG16 model had the best recall score.

5.3.1 Convolutional Neural Network

5.3.1.1 Data preparation, Architecture and Training

First, the dataset is split into training and validation sets. 80% of the 812 images were split into training data (649 images). The split is also stratified. After that split, the validation set is further split into validation and test sets. From the 163 images in the validation set, 70% remained in the validation set (114 images) and 30% in

the test set (49 images). The reason for this split was to provide the training process with more images, as this classification stage is more complex than previous tasks of the pipeline. This way, the model will be able to find meaningful features to predict each class successfully.

The input layer accepts images with 256x256 pixel resolution, with three color channels. It is a 2D convolution with 8 neurons, has a kernel size of (4,4) and stride equal to 1. The parameter "padding" is equal to "same" and it uses ReLU as an activation function. The following layers include a Batch normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Following, another 2D Convolution layer is placed with 16 neurons, kernel size of (4,4), stride equal to 1, padding equal to "same" and ReLU activation function. The following two layers also include a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Later on, another 2D Convolution is added with 32 neurons, a stride of 1, a kernel size of (4,4), padding equal to "same" and ReLU activation function. Also here, the two layers that follow are a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". A Dropout layer is then added, in order to randomly deactivate a portion of the units during each training update, with a probability of 0.1. Then, a 2D Convolution with 16 neurons, kernel size of (4,4), stride equal to 1, padding equal to "same" and ReLU activation function is added. The two layers that follow are also a Batch Normalization layer, a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Right before the fully connected layers, a Flatten layer is added to reshape the data into a 1-dimensional array, making the data to have a valid shape for entering the fully connected layers. Moreover, four Dense layers are added, with 32, 16, 16 and 8 neurons respectively, with all using ReLU as the activation function. Finally, the output layer is also a Dense layer with two neurons, using a sigmoid activation function. More information about this architecture can be seen in Table 16.

Layer	**Output Shape**
Conv2D	(None, 256, 256, 8)
BatchNormalization	(None, 256, 256, 8)
MaxPooling2D	(None, 127, 127, 8)
Conv2D	(None, 127, 127, 16)
BatchNormalization	(None, 127, 127, 16)
MaxPooling2D	(None, 62, 62, 16)

Conv2D	(None, 62, 62, 32)
BatchNormalization	(None, 62, 62, 32)
MaxPooling2D	(None, 30, 30, 32)
Dropout	(None, 30, 30, 32)
Conv2D	(None, 30, 30, 16)
BatchNormalization	(None, 30, 30, 16)
MaxPooling2D	(None, 14, 14, 16)
Flatten	(None, 3136)
Dense	(None, 32)
Dense	(None, 16)
Dense	(None, 16)
Dense	(None, 8)
Dense	(None, 2)

Table 16. Custom CNN model architecture.

During the training process, Early Stopping is also utilized, taking into account the model's validation loss. The training will stop if the validation loss does not improve for five consecutive epochs and the Early Stopping is set to start after epoch 10.

The Adam optimizer is chosen for training. The learning rate is set to 0.0001 and the loss function used is binary cross-entropy. Also, the training is done with a batch size of 32 and the model is going to train for a maximum of 100 epochs. However, due to the Early Stopping, as the validation loss stopped improving, the model stopped training at epoch 42. Finally, the “shuffle” parameter was also set to “True”.

5.3.1.2 Results

The results after the training process ended (shown in Table 17), indicate that the model could not converge. This can be also seen in Figure 36, showing the model’s loss and accuracy across the training process. Finally, the accuracy of the model on the test set was 0.82 and recall 0.70. The confusion matrix in Figure 37 and the classification report in Table 18 shows the model's results in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.9881	0.0662
Validation	0.7456	0.5686

Table 17. Custom CNN model training results.

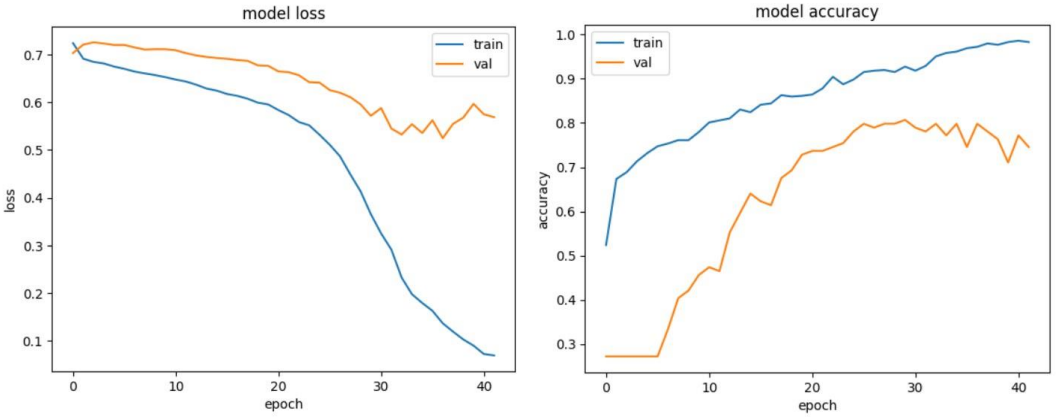


Figure 36. CNN training loss and accuracy. CNN model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

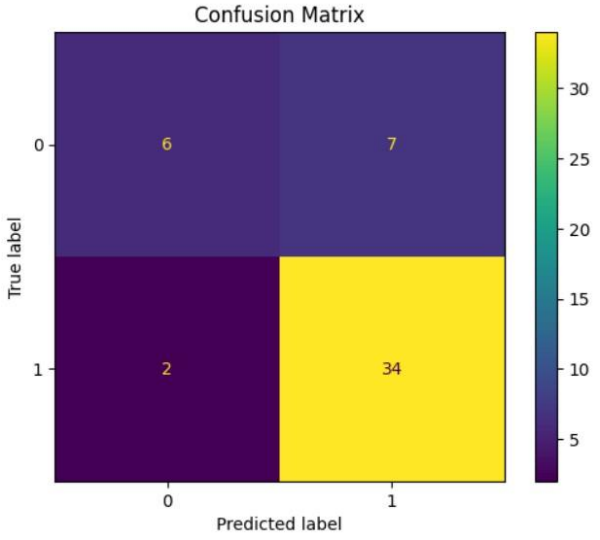


Figure 37. Confusion matrix of the CNN model's predictions on the test set.

	precision	recall	f1-score	support
0	0.75	0.46	0.57	13
1	0.83	0.94	0.88	36
accuracy			0.82	49
macro avg	0.79	0.70	0.73	49

weighted avg	0.81	0.82	0.80	49

Table 18. CNN Classification report.

5.3.2 Convolutional Neural Network with Image Augmentation

5.3.2.1 Data preparation, Architecture and Training

In this experiment, the dataset is split in the same way as in the experiment with the custom CNN above. Therefore, out of the 812 images in total, 649 were used for training, while 114 images formed the validation set and 49 the test set.

An image augmentation module is used to generate more images for the training process (Tensorflow's ImageDataGenerator). The model is built to be deeper than the previously described CNN, as the distinction between the classes is challenging. To begin with, the input layer accepts 256x256 pixel images with three color channels. It is a 2D convolution with 32 neurons, has a kernel size of (4,4) and stride equal to 1. Additionally, the parameter "padding" is equal to "same" and ReLU activation function. The following layers include a Batch normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Following, another 2D Convolution layer is placed with 64 neurons, kernel size of (4,4), stride equal to 1, padding equal to "same" and ReLU activation function. The following two layers also include a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Sequentially, another 2D Convolution is added with 64 neurons, a stride of 1, a kernel size of (4,4), padding equal to "same" and ReLU activation function. The two layers that follow are a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". A 2D Convolution with 64 neurons, kernel size of (4,4), stride equal to 1, padding equal to "same" and ReLU activation function is added. The two layers that follow are also a Batch Normalization layer, a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Another 2D Convolution is added with 32 neurons, a stride of 1, a kernel size of (4,4), padding equal to "same" and ReLU activation function. Also here, the two layers that follow are a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Right before the fully connected layers, a Flatten layer is added to flatten the data. Then, four Dense layers are added, with 128, 64, 32 and 16 neurons respectively, with all using ReLU as the activation function. In these four dense layers, a layer weight regularizer is added. Regularizers allow the application of penalties on each layer parameters, or layer activity during optimization, applied on a per-layer basis. These penalties are summed into the loss function that the network optimizes. More specifically, the L2 regularizer ($L2 = \lambda * \Sigma(w_i^2)$) is chosen, with λ set to 0.01. The L2 regularization, also known as Ridge regularization, adds the sum of the squared values of the model's coefficients to the loss function. This regularization technique does not force the coefficients to be exactly zero but instead encourages them to be small. Also, it can prevent

overfitting by spreading the influence of a single feature across multiple features (Van Otten, 2023). Finally, the output layer is a Dense layer with two neurons, using the sigmoid activation function. A representation of this architecture can be seen in Table 19.

Layer	**Output Shape**
Conv2D	(None, 256, 256, 32)
BatchNormalization	(None, 256, 256, 32)
MaxPooling2D	(None, 127, 127, 32)
Conv2D	(None, 127, 127, 64)
BatchNormalization	(None, 127, 127, 64)
MaxPooling2D	(None, 62, 62, 64)
Conv2D	(None, 62, 62, 64)
BatchNormalization	(None, 62, 62, 64)
MaxPooling2D	(None, 30, 30, 64)
Conv2D	(None, 30, 30, 32)
BatchNormalization	(None, 30, 30, 32)
MaxPooling2D	(None, 14, 14, 32)
Flatten	(None, 6272)
Dense	(None, 128)
Dense	(None, 64)
Dense	(None, 32)
Dense	(None, 16)
Dense	(None, 2)

Table 19. Custom CNN model architecture utilizing Image Augmentation data.

In the training process, Early Stopping is utilized, monitoring the model's validation loss. The training will stop if the validation loss does not improve for five consecutive epochs and is set to start the monitoring after epoch 10.

The images that were generated by the image augmentation library have undergone the following processing:

- Random rotation between -20 and +20 degrees
- Randomly shift the width by a fraction of 0.1
- Randomly shift the height by a fraction of 0.1
- Shear transformations with a maximum shear of 0.1
- Randomly zooming inside images by a fraction of 0.1
- Randomly flip images vertically

The Adam optimizer is chosen for training. The learning rate was set to 0.001 and the loss function used is binary cross-entropy. Also, the training is done with a batch size of 32 and the model is going to train for a maximum of 500 epochs. However, due to the Early Stopping, as the validation loss stopped improving, the model stopped training at epoch 257. Finally, the “shuffle” parameter was set to “True”.

5.3.2.2 Results

The results after the training process was complete (shown in Table 20), indicate that the model overfitted and the model’s accuracy remained flat during the whole training process leading to bad results. This can be also seen in Figure 38, showing the model’s loss and accuracy across the training process. The accuracy of the model on the test set was 0.73 and recall 0.50. The confusion matrix in Figure 39 and the classification report in Table 21 shows the model's results in more detail.

Data subset	**Accuracy**	**Loss**
Training	1.0000	0.1266
Validation	0.7281	0.7637

Table 20. Custom CNN model utilizing Image Augmentation training results.

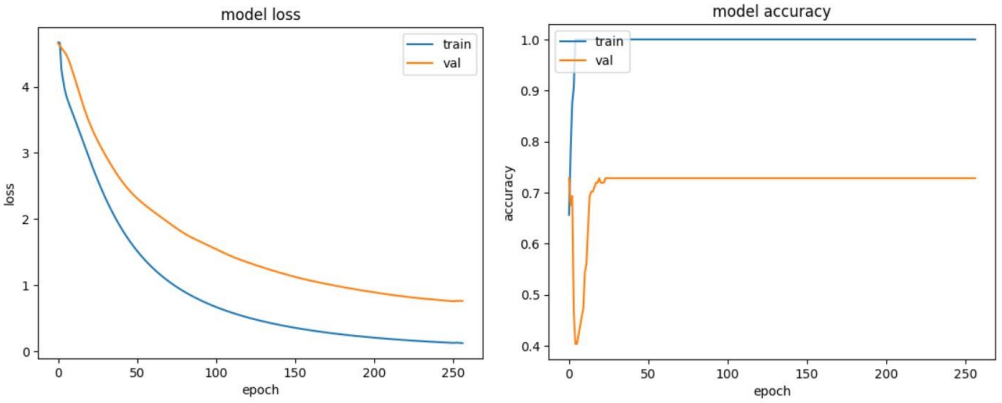


Figure 38. CNN with Image Augmentation training loss and accuracy. Custom CNN model utilizing Image Augmentation results. Loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

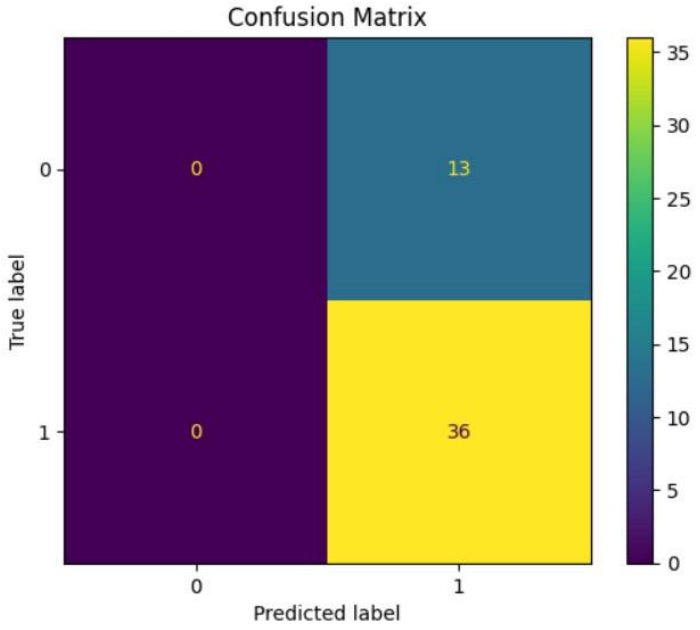


Figure 39. Confusion matrix of the model's predictions on the test set.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	13
1	0.73	1.00	0.85	36
accuracy			0.73	49
macro avg	0.37	0.50	0.42	49
weighted avg	0.54	0.73	0.62	49

Table 21. CNN utilizing Image Augmentation Classification report.

5.3.3 VGG16 Model

This experiment yielded the best results amongst the four models that were implemented for this classification stage.

5.3.3.1 Data preparation, Architecture and Training

In this experiment, the dataset is split the same way as the other models in this stage (649 training images, 114 validation images and 49 test images). The VGG16

model is used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false and the model's layers are frozen.

After the VGG16 base model's architecture, several custom layers are added. The input layer accepts images with a shape of 256x256 pixels with three color channels. The following layers include a Global Average Pooling layer and a Batch Normalization layer, followed by a Flatten layer. Later, five Dense layers are added with ReLU as the activation function, with each one having 128, 64, 32, 16 and 8 neurons respectively. Finally, the output layer is a Dense layer with two neurons, using the sigmoid activation function (Figure 40).

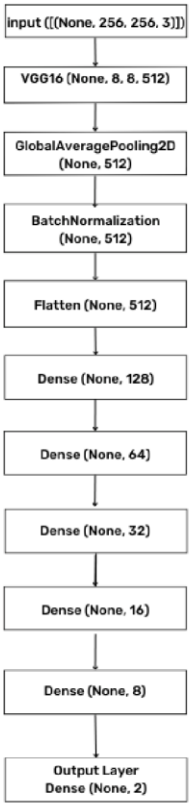


Figure 40. VGG16 model architecture.

Early Stopping is also utilized in this model, monitoring its validation loss during training. The training thus stops in case the validation loss does not improve for five consecutive epochs. It is also set to start monitoring after 10 epochs.

The Adam optimizer is used for training. The learning rate was set to 0.0001 and the loss function used is binary cross-entropy. Additionally, the training is done with a batch size of 32 and the model is going to train for a maximum of 100 epochs. As the validation loss of this model stopped improving, the training halted at epoch 41. Last but not least, the "shuffle" parameter was set to "True".

5.3.3.2 Results

The results of the training process (shown in Table 22), indicate that the model's training was stable, but the loss diagram indicates that the model could be prone to overfitting, thus deeper model architectures were avoided. This can be also seen in

Figure 41, showing the model's loss and accuracy across the training process. Also, the accuracy of the model on the test set was 0.94 and recall 0.91. The confusion matrix in Figure 42 and the classification report in Table 23 shows the model's results in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.9832	0.0548
Validation	0.8947	0.2484

Table 22. VGG16 model training results.

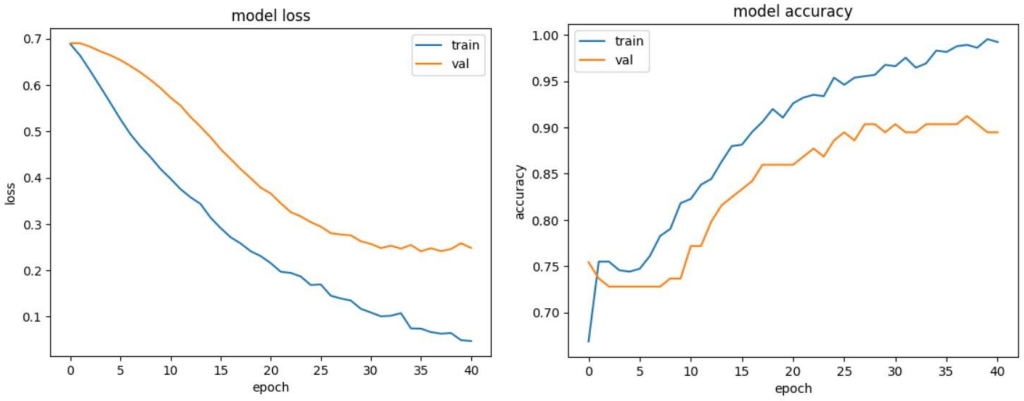


Figure 41. VGG16 training loss and accuracy. VGG16 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

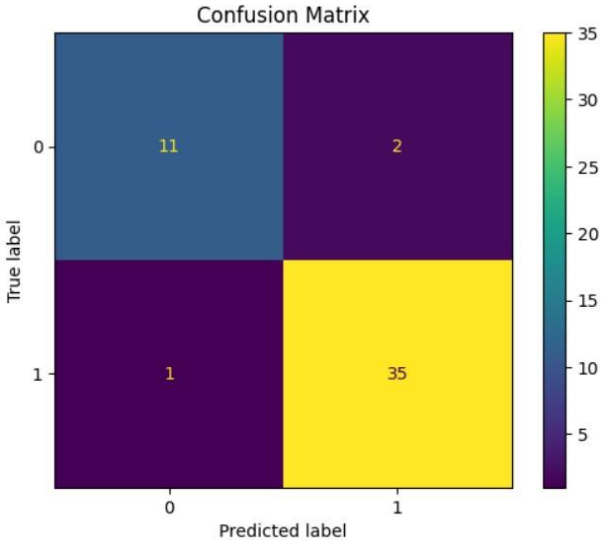


Figure 42. Confusion matrix of the VGG16 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.92	0.85	0.88	13
1	0.95	0.97	0.96	36

accuracy			0.94	49
macro avg	0.93	0.91	0.92	49
weighted avg	0.94	0.94	0.94	49

Table 23. VGG16 Classification report.

5.3.4 ResNet50 Model

5.3.4.1 Data preparation, Architecture and Training

In this experiment, the dataset is split as described before, using 649 images for training, 114 images for validation and 49 images for testing. The model is used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false and the model's layers are frozen.

After the ResNet50 base model's architecture, several custom layers are introduced. The input layer accepts 256x256 images with three color channels. The layers that follow include a Global Average Pooling layer and a Batch Normalization layer, as well as a Flatten layer. Later, four Dense layers are present with ReLU activation function, each having 32, 32, 16 and 8 neurons respectively. Finally, the output layer is a Dense layer with two neurons, using sigmoid activation function (Figure 43).

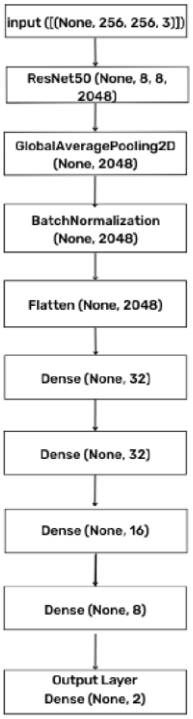


Figure 43. ResNet50 model architecture.

The Early Stopping technique is used in this model, monitoring the training process progress, based on the validation loss. The training will stop if the validation loss does

not improve for five consecutive epochs, while Early Stopping is set to begin monitoring after 10 epochs.

Adam optimizer is utilized for training. The learning rate is set to be equal to 0.001, and the loss function used is binary cross-entropy. Additionally, the training is conducted using a batch size of 16, with the "shuffle" parameter set to "True".

The model is expected to train for a maximum of 100 epochs. However, due to the early stopping mechanism, and the fact that the validation loss of this model stopped improving, the process halted at epoch 18.

5.3.4.2 Results

The final results after the training process (shown in Table 24), indicate that this model also started to overfit. This can be seen in Figure 44, showing the model's loss and accuracy during the training process. The accuracy of the model on the test set was 0.86 and recall 0.83. The confusion matrix in Figure 45 and the classification report in Table 25 shows the model's results in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.9264	0.2087
Validation	0.7719	0.5609

Table 24. ResNet50 model training results.

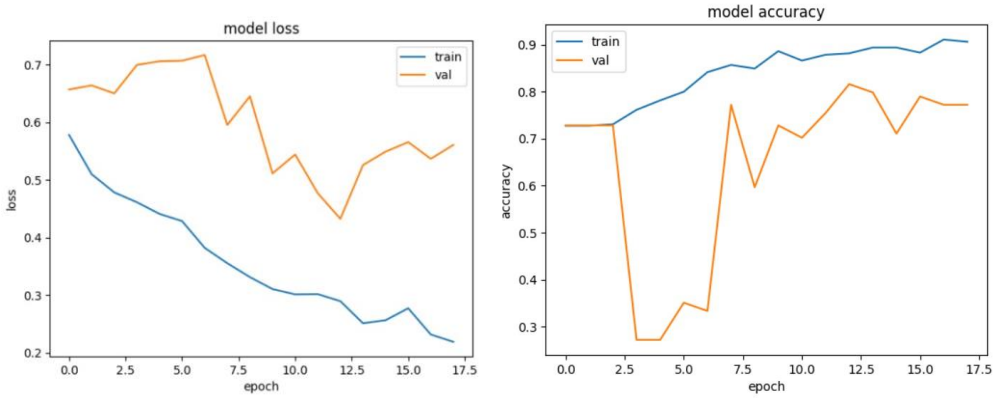


Figure 44. ResNet50 training loss and accuracy. ResNet50 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

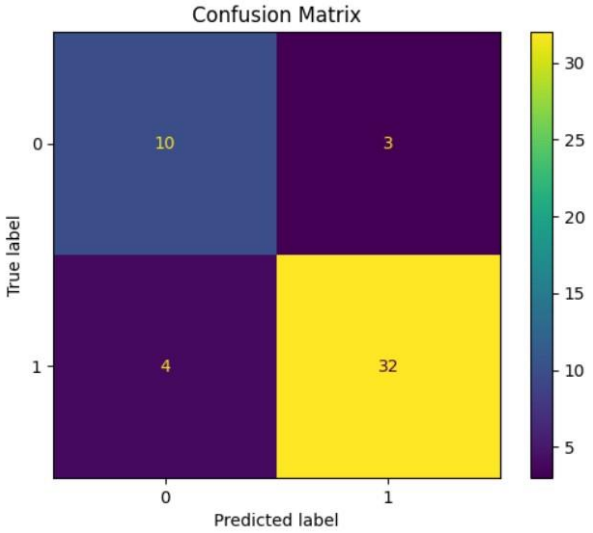


Figure 45. Confusion matrix of the ResNet50 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.71	0.77	0.74	13
1	0.91	0.89	0.90	36
accuracy			0.86	49
macro avg	0.81	0.83	0.82	49
weighted avg	0.86	0.86	0.86	49

Table 25. ResNet50 Classification report.

5.4 Arthroplasty / Nailing Classification Stage

This is the fourth stage of the classification pipeline. Aim of this phase is to classify x-rays of operations, to arthroplasty or nailing. For this purpose, various models were trained and evaluated, with the VGG16 model achieving the best accuracy.

5.4.1 Convolutional Neural Network

5.4.1.1 Data preparation, Architecture and Training

In this classification stage, certain data were chosen to be used for the training process. More specifically, the classes that were encompassed in the dataset of this stage are:

- Left arthroplasty
- Right arthroplasty
- Left nailing
- Right nailing

Left and right arthroplasty x-rays are merged in one class (arthroplasty), and left and right nailing x-rays in another class (nailing). This resulted in 221 images in total. Then 60% of these images were split into training data (132 images). The split

is stratified. After that split, the validation set is further split into validation and test sets. From the 89 images in the validation set, 65% of the images remained in the validation set (57 images) and 35% in the test set (32 images).

Different architectures were tried for this custom CNN, but the deeper the model was, the easier it was for the model to overfit. This led the adopted architecture to be relatively shallow. To begin with, the input layer receives 256x256 pixel images with three color channels. It is a 2D convolution with 8 neurons, kernel size of (4,4) and stride equal to 1. The parameter "padding" is equal to "same" and it uses ReLU activation function. The following layers include a Batch normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Following, another 2D Convolution layer is placed with 16 neurons, kernel size of (4,4), stride equal to 1, padding equal to "same" and ReLU activation function. The following two layers also include a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Later, another 2D Convolution is added with 8 neurons, a stride of 1, a kernel size of (4,4), padding equal to "same" and using ReLU activation function. Also here, the two layers that follow are a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Next, a Flatten layer is added to reshape the data, so that the data are flattened to enter the fully connected layers. Four Dense layers are then added, with 64, 32, 16 and 8 neurons respectively, all using the ReLU activation function. Finally, the output layer is a Dense layer with two neurons, using a sigmoid activation function. A representation of this architecture can be seen in Table 26.

Layer	**Output Shape**
Conv2D	(None, 256, 256, 8)
BatchNormalization	(None, 256, 256, 8)
MaxPooling2D	(None, 127, 127, 8)
Conv2D	(None, 127, 127, 16)
BatchNormalization	(None, 127, 127, 16)
MaxPooling2D	(None, 62, 62, 16)
Conv2D	(None, 62, 62, 8)
BatchNormalization	(None, 62, 62, 8)
MaxPooling2D	(None, 30, 30, 8)

Flatten	(None, 7200)
Dense	(None, 64)
Dense	(None, 32)
Dense	(None, 16)
Dense	(None, 8)
Dense	(None, 2)

Table 26. Custom CNN model architecture.

Early Stopping is utilized during the training of the model, to monitor the validation loss. The training will stop automatically in case the validation loss does not improve for five consecutive epochs. The Early Stopping is set to start monitoring after the 10th epoch.

The Adam optimizer is being used for training. The learning rate is 0.0001 and the loss function used is binary cross-entropy. The training is done with a batch size of 32 and the model is going to train for a maximum of 100 epochs. However, as the validation loss stopped improving, the model stopped training at epoch 56. Finally, the "shuffle" parameter was also set to "True" to shuffle the data.

5.4.1.2 Results

The training results shown in Table 27, indicate that the CNN model overfitted. This can be seen in Figure 46, depicted on the model's loss (left image). Finally, the accuracy of the model on the test set was 0.72. The confusion matrix in Figure 47 and the classification report in Table 28 shows the model's predictions in more detail. These results indicate that the model was not able to learn how to discriminate between the two classes, and one reason may be the limited number of available data for this task.

Data subset	**Accuracy**	**Loss**
Training	1.0000	0.0098
Validation	0.7193	0.6242

Table 27. Custom CNN model training results.

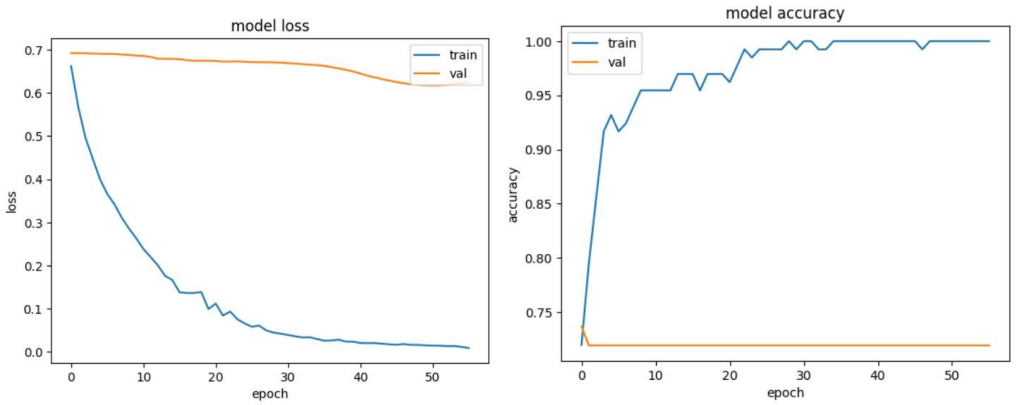


Figure 46. Custom CNN training loss and accuracy. Custom CNN model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

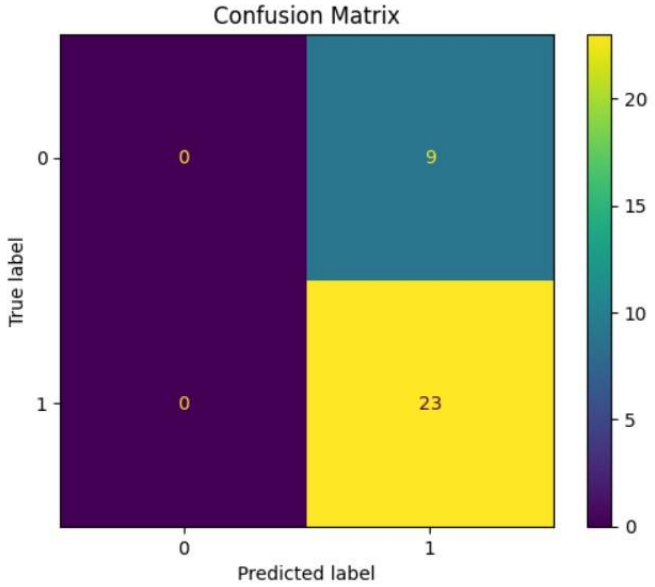


Figure 47. Confusion matrix of the CNN model's predictions on the test set.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	9
1	0.72	1.00	0.84	23
accuracy			0.72	32
macro avg	0.36	0.50	0.42	32
weighted avg	0.52	0.72	0.60	32

Table 28. Custom CNN Classification report.

5.4.2 ResNet50 Model

5.4.2.1 Data preparation, Architecture and Training

In this experiment, the dataset is split like above, with 132 training images, 57 validation images and 32 test images. The ResNet50 model is used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false and the model's layers are frozen.

After the ResNet50 base model's architecture, several custom layers are added. The input layer takes 256x256 pixel images with three color channels. The following layers include a Global Average Pooling layer and a Batch Normalization layer, followed by a Flatten layer. Then, four Dense layers are added with ReLU activation function, with 32, 32, 16 and 8 neurons each, in this exact order. Finally, the output layer is a Dense layer with two neurons, using a sigmoid activation function (Figure 48).

Figure 48. ResNet50 model architecture.

Early Stopping is applied to the process of training the model, monitoring the validation loss. The training is set to stop in case the validation loss does not improve for five consecutive epochs. Additionally, the Early Stopping is activated after 10 epochs.

The Adam optimizer is selected for training. The learning rate was set to 0.001 and the loss function used is binary cross-entropy. To elaborate, during the training a batch size of 16 is being used, and the model is going to train for a maximum of 100 epochs. Due to the fact that the validation loss stopped improving, the training came to a halt at epoch 16. Finally, the "shuffle" parameter was set to "True".

5.4.2.2 Results

The results, as shown in Table 29, indicate that this model also overfitted. This can be also seen in Figure 49, showing the model's loss and accuracy across the training process. The accuracy of the model on the test is 0.72. The confusion matrix in Figure 50 and the classification report in Table 30 shows the model's predictions in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.9478	0.1390
Validation	0.7193	1.2450

Table 29. ResNet50 model training results.

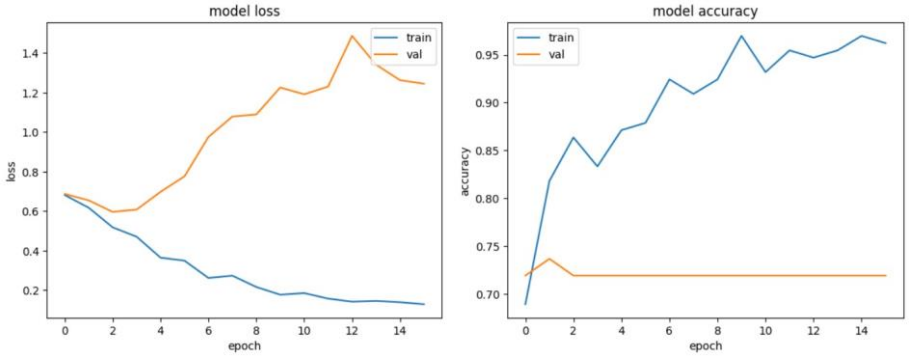


Figure 49. ResNet50 training loss and accuracy. ResNet50 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

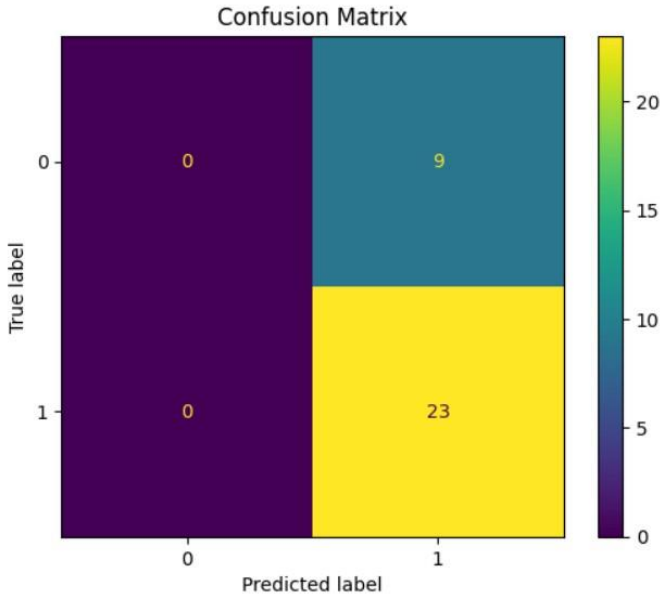


Figure 50. Confusion matrix of the ResNet50 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	9

1	0.72	1.00	0.84	23
accuracy			0.72	32
macro avg	0.36	0.50	0.42	32
weighted avg	0.52	0.72	0.60	32

Table 30. ResNet50 Classification report.

5.4.3 VGG16 Model

This experiment had the best results amongst the three models that were implemented for this classification stage.

5.4.3.1 Data preparation, Architecture and Training

In this experiment, the dataset is split as described above, with 132 images for training, 57 images for validation and 32 images for testing. The VGG16 model is used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false and the model's layers are frozen.

After the VGG16 base model's architecture, several custom layers are added. The input layer takes 256x256 pixel images with three color channels as an input. The following layers include a Global Average Pooling layer and a Batch Normalization layer, followed by a Flatten layer. Later, five Dense layers are added with the ReLU activation function, with each one having 128, 64, 32, 16 and 8 neurons respectively. Finally, the output layer is a Dense layer with two neurons, using a sigmoid activation function (Figure 51).

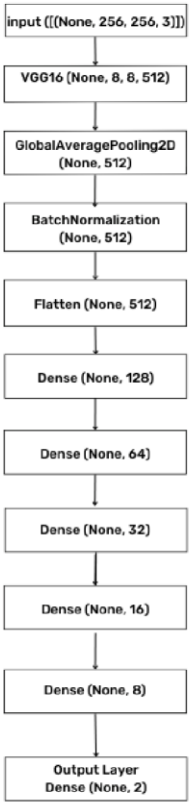


Figure 51. VGG16 model architecture.

Early Stopping is utilized during the training of the model, monitoring its validation loss. The patience parameter is set to five epochs and the mechanism is set to start monitoring after 10 epochs.

The Adam optimizer is chosen for training. The learning rate was set to 0.0001 and the loss function used is binary cross-entropy. Additionally, the training is done with a batch size of 32 and the model is going to train for a maximum of 500 epochs. Nonetheless, the validation loss of this model stopped improving and the training stopped at epoch 123. Finally, the "shuffle" parameter was set to "True".

5.4.3.2 Results

Based on the training results shown in Table 31, it can be concluded that the model's training was stable. This can also be seen in Figure 52, showing the model's loss, as well as the accuracy throughout the training process. Finally, the accuracy of the model on the test set is 1.0. The confusion matrix in Figure 53 and the classification report in Table 32 shows the model's results in more detail.

Data subset	Accuracy	Loss
Training	1.0000	0.0070
Validation	0.9649	0.0681

Table 31. VGG16 model training results.

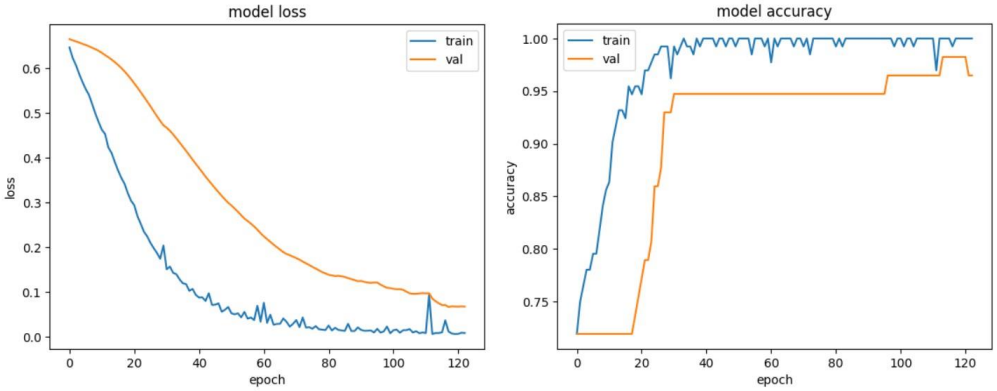


Figure 52. VGG16 training loss and accuracy. VGG16 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

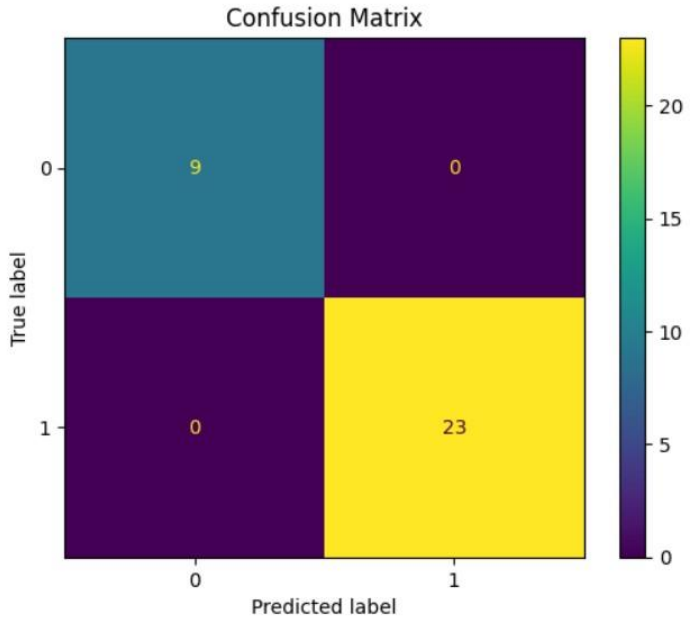


Figure 53. Confusion matrix of the VGG16 model's predictions on the test set.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	23
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

Table 32. VGG16 Classification report.

5.5 Subcapital Fracture / Intertrochanteric Fracture / Osteoarthritis Classification Stage

This is the fifth and final stage of the classification pipeline. In this step, the aim is to classify x-rays to three classes: subcapital fracture, intertrochanteric fracture and osteoarthritis. Various models were trained and their performance was measured, with the VGG16 model having the best results.

5.5.1 Convolutional Neural Network

5.5.1.1 Data preparation, Architecture and Training

In this classification stage, certain data were chosen for the training process. More specifically, the classes that were used for this task are:

- Left intertrochanteric
- Right intertrochanteric
- Left subcapital
- Right subcapital
- Left osteoarthritis
- Right osteoarthritis

The data are grouped into three classes: intertrochanteric fracture, subcapital fracture and osteoarthritis. This resulted in forming a dataset specifically tailored for this task, consisting of 385 images in total. After the first split, 65% of these images were used as the training data (250 images), and the split is stratified. After that split, the validation set is further split into validation and test sets. From the 135 images in the validation set, 70% of the images remained in the validation set (94 images) and 30% in the test set (41 images).

The input layer is tailored for 256x256 pixel images with three color channels. It is a 2D convolution with 32 neurons, has a kernel size of (4,4) and stride equal to 1. The parameter “padding” is equal to “same” and it uses ReLU activation function. The following layers include a Batch normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to “valid”. Following, another 2D Convolution layer is placed with 64 neurons, kernel size of (4,4), stride equal to 1, padding equal to “same” and ReLU activation function. The following two layers also include a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to “valid”. Right after, another 2D Convolution is added with 128 neurons, a stride of 1, a kernel size of (4,4), padding equal to “same” and using ReLU activation function. The two layers that follow are a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to “valid”. Another 2D Convolution layer is then added with 64 neurons, a stride of 1, a kernel size of (4,4), padding equal to “same” and using ReLU activation function. The two layers that follow are also a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to “valid”. Before the Dense layers, a final 2D Convolution is added with 32 neurons, a stride of 1, a kernel size of (4,4), padding equal to “same” and using ReLU

activation function. The two layers after that are a Batch Normalization layer and a 2D Max Pooling layer, with a kernel size of (4,4), stride equal to 2 and padding equal to "valid". Furthermore, a Flatten layer is added to transform the data to a 1-dimensional array, so that the data can enter the fully connected layers. These layers are six Dense layers, with 128, 64, 32, 32, 16 and 8 neurons respectively, all using the ReLU activation function. Finally, the output layer is a Dense layer of three neurons, using a softmax activation function. A representation of this architecture can be seen in Table 33.

Layer	**Output Shape**
Conv2D	(None, 256, 256, 32)
BatchNormalization	(None, 256, 256, 32)
MaxPooling2D	(None, 127, 127, 32)
Conv2D	(None, 127, 127, 64)
BatchNormalization	(None, 127, 127, 64)
MaxPooling2D	(None, 62, 62, 64)
Conv2D	(None, 62, 62, 128)
BatchNormalization	(None, 62, 62, 128)
MaxPooling2D	(None, 30, 30, 128)
Conv2D	(None, 30, 30, 64)
BatchNormalization	(None, 30, 30, 64)
MaxPooling2D	(None, 14, 14, 64)
Conv2D	(None, 14, 14, 32)
BatchNormalization	(None, 14, 14, 32)
MaxPooling2D	(None, 6, 6, 32)
Flatten	(None, 1152)
Dense	(None, 128)

Dense	(None, 64)
Dense	(None, 32)
Dense	(None, 32)
Dense	(None, 16)
Dense	(None, 8)
Dense	(None, 3)

Table 33. Custom CNN model architecture.

Early Stopping is utilized in the training process, monitoring the model's validation loss after the 10th epoch. The training will therefore stop in case the validation loss does not decrease for five consecutive epochs.

The Adam optimizer is utilized during training. The learning rate is set to 0.001 and the loss function used is “categorical cross-entropy”. Also, the training is executed using a batch size of 16 and the model is designed to train for 100 epochs. However, due to the Early Stopping mechanism functionality, the model stopped training at epoch 24. Finally, the “shuffle” parameter was set to “True”.

5.5.1.2 Results

After the completion of the training process, the results as shown in Table 34, indicate that the CNN model had fluctuations and could not converge. This can be seen in Figure 54, depicted on the model’s loss and accuracy. Finally, the model’s accuracy on the test set was 0.66. The confusion matrix in Figure 55 and the classification report in Table 35 shows the model's predictions in more detail.

Data subset	**Accuracy**	**Loss**
Training	0.8600	0.3314
Validation	0.6596	1.4093

Table 34. Custom CNN model training results.

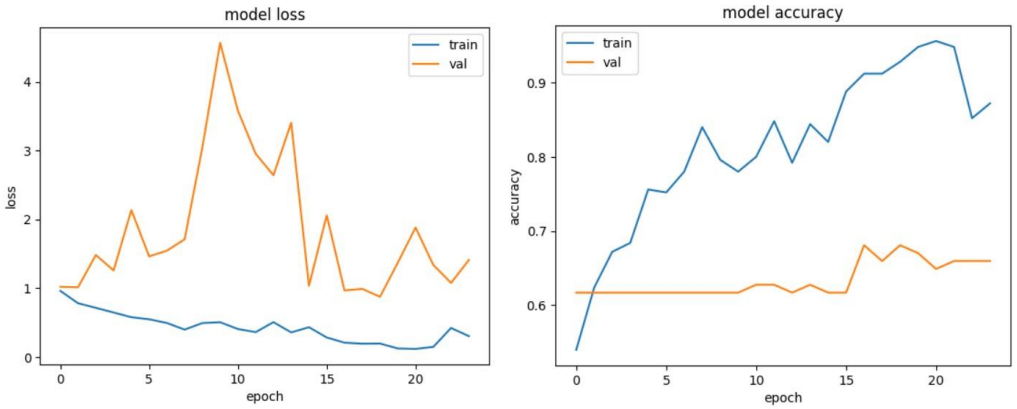


Figure 54. Custom CNN training loss and accuracy. Custom CNN model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

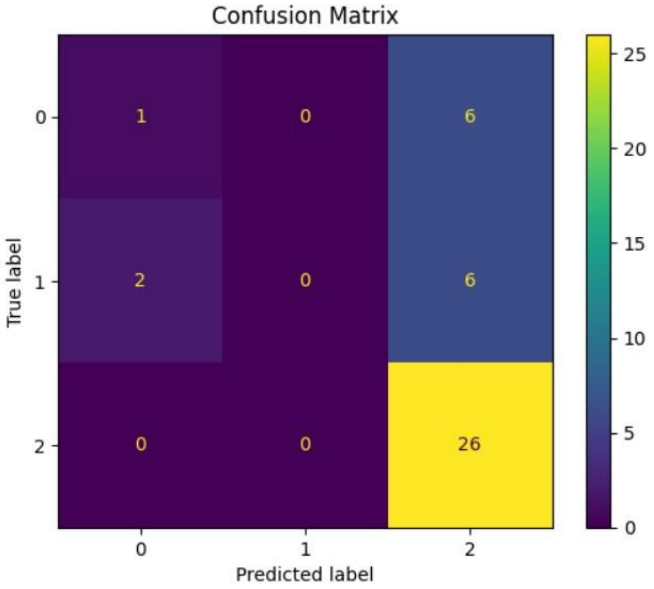


Figure 55. Confusion matrix of the CNN model's predictions on the test set.

	precision	recall	f1-score	support
0	0.33	0.14	0.20	7
1	0.00	0.00	0.00	8
2	0.68	1.00	0.81	26
accuracy			0.66	41
macro avg	0.34	0.38	0.34	41
weighted avg	0.49	0.66	0.55	41

Table 35. Custom CNN Classification report.

5.5.2 ResNet50 Model

5.5.2.1 Data preparation, Architecture and Training

In this experiment, the dataset is split as mentioned above (250 training images, 94 validation images and 41 test images). The ResNet50 model is used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false, while the model's layers are frozen.

After the ResNet50 base model's architecture, several custom layers are added. The input layer receives 256x256 pixel images with three color channels. The next layers include a Global Average Pooling layer and a Batch Normalization layer, followed by a Flatten layer. Later, four Dense layers are added with ReLU activation function, having 32, 32, 16 and 8 neurons each. Finally, the output layer is a Dense layer with three neurons, using a softmax activation function (Figure 56).

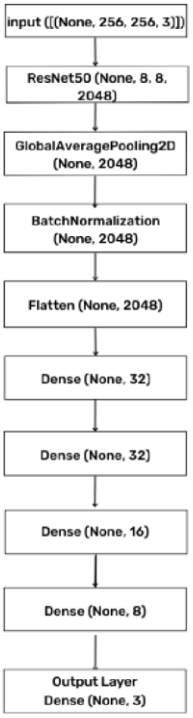


Figure 56. ResNet50 model architecture.

Early Stopping technique is also adopted in training this model, for monitoring its validation loss during training. The training is designated to stop if the validation loss does not improve for five consecutive epochs. Also, the early stopping is set to start monitoring after the completion of the first 10 epochs.

The Adam optimizer is used for training, while the learning rate is equal to 0.001 and the loss function is "categorical cross-entropy". The training is conducted with a batch size of 16 and the model is set to train for a maximum of 100 epochs. Due to the progress of the validation loss throughout the training, Early Stopping terminated the training at epoch 16. At last, the "shuffle" parameter is set to "True".

5.5.2.2 Results

The results, shown in Table 36, indicate that the model overfitted. This can also be concluded from Figure 57, showing the model's loss and accuracy during the training process. The model scored an accuracy of 0.66 on the test set. The confusion matrix in Figure 58 and the classification report in Table 37 shows the model's results in more detail.

Data subset	Accuracy	Loss
Training	0.8939	0.3073
Validation	0.6277	1.1889

Table 36. ResNet50 model training results.

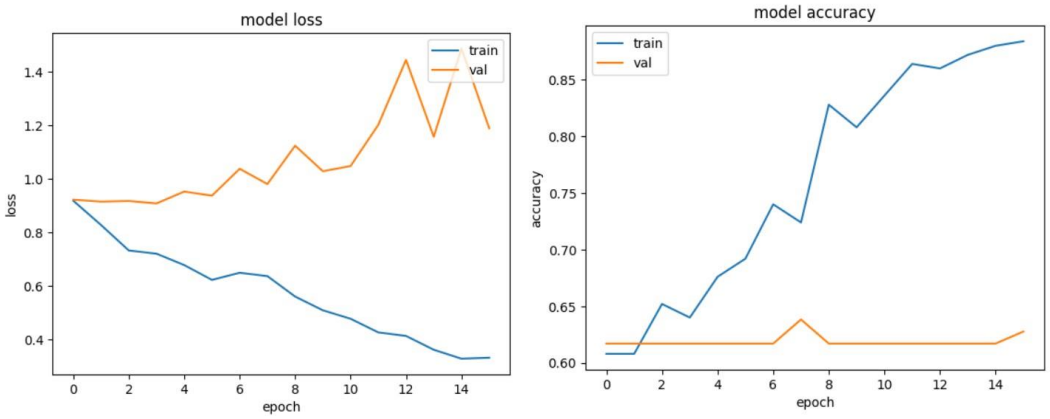


Figure 57. ResNet50 training loss and accuracy. ResNet50 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

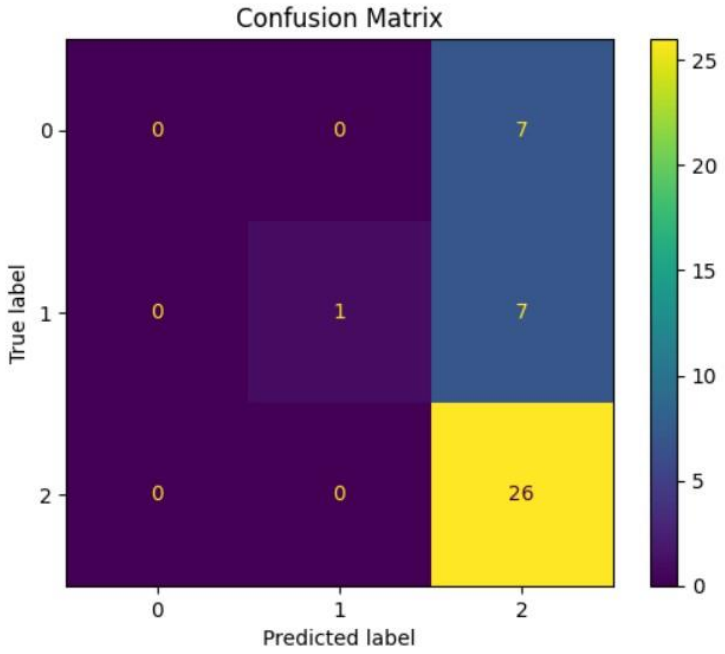


Figure 58. Confusion matrix of the ResNet50 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	7
1	1.00	0.12	0.22	8
2	0.65	1.00	0.79	26
accuracy			0.66	41
macro avg	0.55	0.38	0.34	41
weighted avg	0.61	0.66	0.54	41

Table 37. ResNet50 Classification report.

5.5.3 VGG16 Model

This experiment yielded the best results amongst the three models that were implemented for this classification stage.

5.5.3.1 Data preparation, Architecture and Training

In this experiment, the dataset split is implemented as described above (250 training images, 94 validation images and 41 test images). The VGG16 model is used with its pre-trained weights from the ImageNet dataset. The parameter *"include_top"* is set to false and the model's layers are frozen.

Following the VGG16 base model's architecture, several custom layers are added. The input layer is designed to accept 256x256 pixel images with three color channels. The next layers include a Global Average Pooling layer and a Batch Normalization layer, as well as a Flatten layer. Continuing, two Dense layers are added with ReLU activation function, with each one having 256 and 128 neurons respectively. Finally, the output layer is a Dense layer with three neurons, using softmax as the activation function (Figure 58).

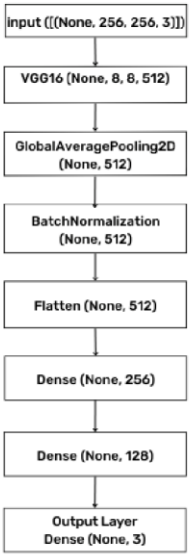


Figure 58. VGG16 model architecture.

Early Stopping is utilized in this experiment, monitoring the validation loss during training. The training will be terminated in case the validation loss does not improve for five consecutive epochs and the monitoring is set to start after the first 10 epochs.

The Adam optimizer is utilized during training. The learning rate is set to 0.001 and the loss function used is "categorical cross-entropy". Additionally, the training is done with a batch size of 16 and the model is going to train for a maximum of 500 epochs. Due to the early stopping mechanism, the training stopped at epoch 28. Last but not least, the "shuffle" parameter is set to "True".

5.5.3.2 Results

The results after the training process ended (shown in Table 38), indicate that the model could not converge properly. Several changes were made to the model's architecture to overcome this issue, with this architecture achieving the best results. This can be also seen in Figure 59, showing the model's loss and accuracy across the training process. Also, the accuracy of the model on the test set has an accuracy of 0.73. The confusion matrix in Figure 60 and the classification report in Table 39 shows the model's predictions in more detail.

Data subset	**Accuracy**	**Loss**
Training	1.0000	0.0020
Validation	0.7766	0.7211

Table 38. VGG16 model training results.

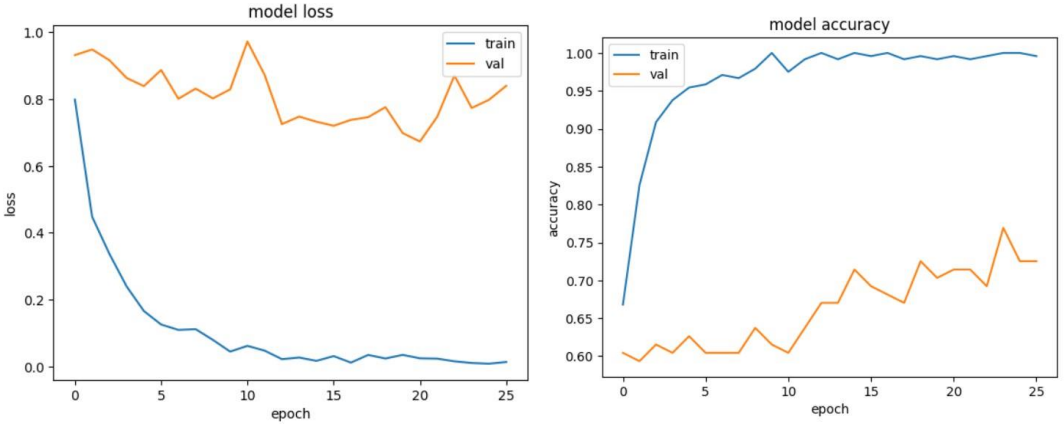


Figure 59. VGG16 training loss and accuracy. VGG16 model loss is depicted on the left image and accuracy on the right. The yellow line represents the validation set, while the blue line represents the training set.

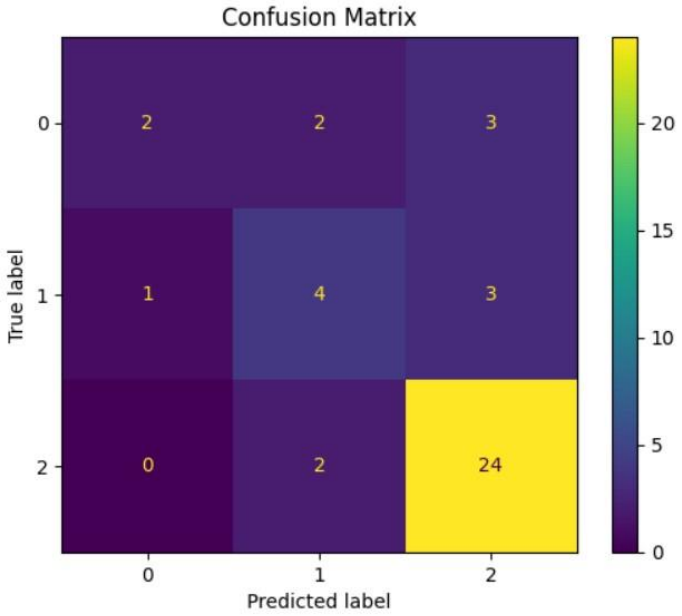


Figure 60. Confusion matrix of the VGG16 model's predictions on the test set.

	precision	recall	f1-score	support
0	0.67	0.29	0.40	7
1	0.50	0.50	0.50	8
2	0.80	0.92	0.86	26
accuracy			0.73	41
macro avg	0.66	0.57	0.59	41

weighted avg	0.72	0.73	0.71	41

Table 39. VGG16 Classification report.

5.6 Conditional Generative Adversarial Network

With the purpose of improving each models' accuracy and making the dataset more balanced, a conditional Generative Adversarial Network (GAN) was implemented to generate more images for each class and ultimately add more images to the dataset.

5.6.1 Data Preprocessing

Firstly, the dataset is preprocessed before it is fed into the GAN architecture. The data preprocessing steps include resizing them to a fixed size and normalizing pixel values to be within the range [0, 1], by dividing by 255. More specifically the images are resized to 128x128 pixels for consistency across the dataset. This size of images was the biggest that could be chosen, due to limitations of the available RAM in Google Colab.

5.6.2 Discriminator Network

Role of the discriminator network is to distinguish between real and fake images while considering the class labels. The discriminator model incorporates both image inputs and label inputs via an embedding layer. In the discriminator, class labels are passed through an embedding layer that maps each label to a 50-dimensional vector. This embedded label is reshaped and concatenated with the input image along a new channel dimension. The combined data is processed through several convolutional layers, followed by Leaky ReLU activation and dropout for regularization. The final output is a single node with a sigmoid activation function, classifying the input as either real or fake. The network's architecture is depicted in more detail in Table 40.

Layer	**Output Shape**
Input	[(None,1)]
Embedding	(None, 1, 50)
Dense	(None, 1, 16384)
Input	(None, 128, 128, 3)
Reshape	(None, 128, 128, 1)
Concatenate	(None, 128, 128, 4)
Conv2D	(None, 64, 64, 256)

Leaky_ReLU	(None, 64, 64, 256)
Conv2D	(None, 32, 32, 128)
Leaky_ReLU	(None, 32, 32, 128)
Flatten	(None, 131072)
Dropout(0.4)	(None, 131072)
Dense	(None, 1)

Table 40. Discriminator network architecture.

5.6.3 Generator Network

The generator model is responsible for creating synthetic images from random noise and conditioned on a class label. The generator thus takes two inputs: random noise (latent space) and a class label. The label is embedded and reshaped, similar to the discriminator, and then concatenated with the latent vector, which is passed through multiple transposed convolutional layers (Conv2DTranspose) to progressively upsample the image. The final output layer generates a 128x128 color image with pixel values in the range [-1, 1], using the tanh activation function. Its architecture can be seen in more detail in Table 41.

Layer	**Output Shape**
Input	[(None,120)]
Input	[(None,1)]
Dense	(None,8192)
Embedding	(None, 1, 50)
Leaky_ReLU	(None, 8192)
Dense	(None, 1, 64)
Reshape	(None, 8, 8, 128)
Reshape	(None, 8, 8, 1)
Concatenate	(None, 8, 8, 129)

Conv2D_Transpose	(None, 16, 16, 512)
Leaky_ReLU	(None, 16, 16, 512)
Conv2d_Transpose	(None, 32, 32, 1024)
Leaky_ReLU	(None, 32, 32, 1024)
Conv2D_Transpose	(None, 64, 64, 512)
Leaky_ReLU	(None, 64, 64, 512)
Conv2D_Transpose	(None, 128, 128, 128)
Leaky_ReLU	(None, 128, 128, 128)
Conv2D	(None, 128, 128, 3)

Table 41. Generator network architecture.

5.6.4 GAN Model

The GAN model combines both the generator and discriminator models. The generator outputs synthetic images that are passed directly into the discriminator for classification. Since the discriminator is pre-trained separately, it is frozen during GAN training, and only the generator's weights are updated based on the discriminator's feedback.

Additionally, the training procedure alternates between training the discriminator and the generator. The training loop is structured to update the discriminator using real and fake samples and update the generator using the feedback from the discriminator. The generator's goal is to reduce the GAN loss, thus generating images that the discriminator cannot distinguish from real ones. Finally, the training process runs for 500 epochs.

In order to generate fake samples, random noise vectors and corresponding labels are created. These are passed into the generator to produce synthetic images. This ensures that the GAN is capable of generating class-conditioned images by providing both noise and labels as inputs to the generator.

5.6.5 Results

When the models finished training, the discriminator's loss on distinguishing real images from the dataset is 0.580. A lower value indicates that the discriminator is performing well in classifying real images correctly. However, if it gets too low, the discriminator could be overfitting to the real images, reducing the overall effectiveness of GAN training. Also, the discriminator’s loss when classifying fake images is 0.559. A lower loss means the discriminator is effectively detecting that the fake images are

not real, while a higher loss could suggest the generator is improving and producing more realistic images. Finally, the generator's goal is to minimize its loss by fooling the discriminator into classifying fake images as real. A loss of 1.020 indicates that the generator is moderately successful in producing convincing fake images, though there is still room for improvement. To summarize, the discriminator has moderately balanced losses for both real and fake images (d1 = 0.580 and d2 = 0.559), which suggests that it is functioning as expected and can distinguish between real and generated images with some confidence. The generator's loss (g = 1.020) indicates that it is producing images that are somewhat realistic, but it still faces challenges in fully deceiving the discriminator. The network's losses are also shown in Figure 61.

When the model stopped training, the losses show that both the generator and discriminator are improving, and the GAN is moving towards producing more realistic images while maintaining a competitive balance between the two models. However, further epochs might still be needed for optimal results or using higher quality images for the training process.

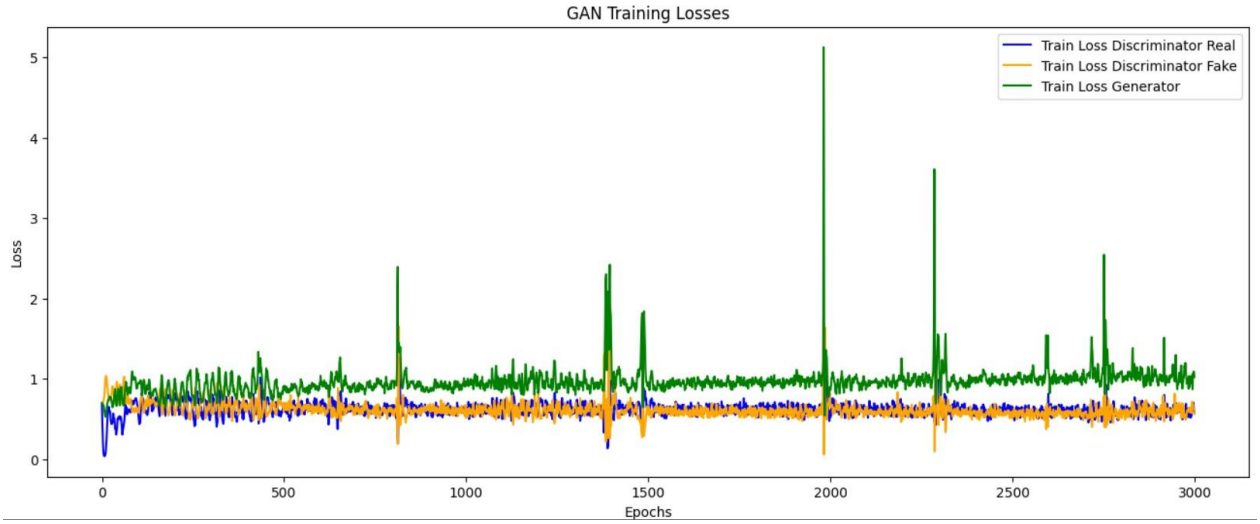
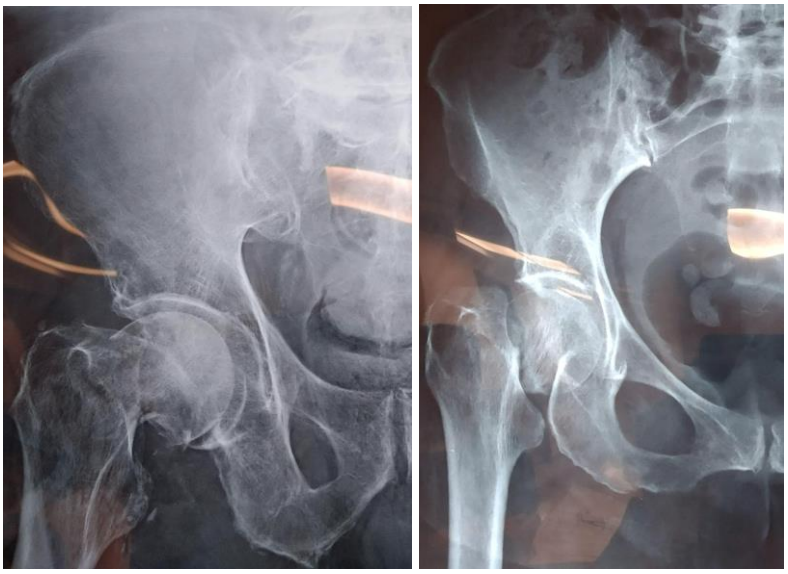


Figure 61. Conditional GAN's training losses.

The conditional GAN's generated images are shown in Figure 63. Due to noise in the training data (e.g. "flares" from light reflecting on the x-ray images, see Figure 62), the GAN generated some orange spots. This problem could be tackled by acquiring more data, or higher quality data. Although this method for generating medical images is state-of-the-art, the generated data do not conform with real data.



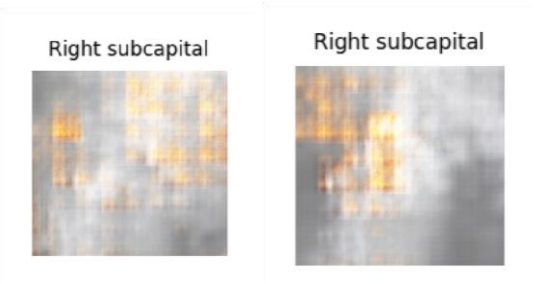


Figure 62. Light reflecting on the images on the two images above. Below are two images generated by the GAN.

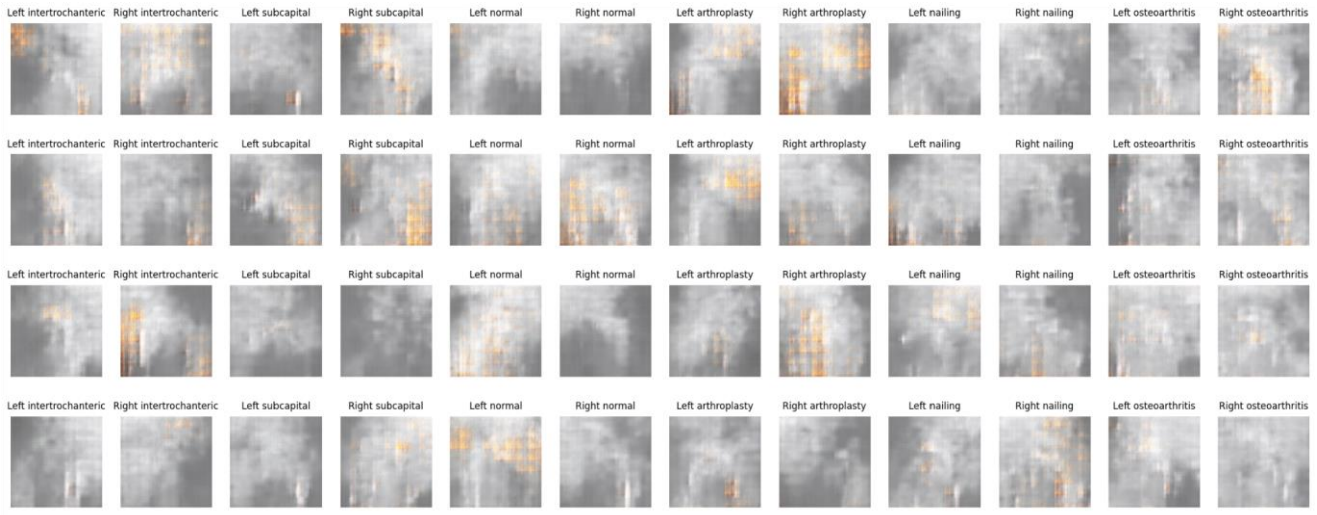


Figure 63. Images generated by the conditional GAN for each class.

6 Chapter 6: Final Pipeline

Amongst the experiments of the classification stages, the following models were chosen for the classification pipeline based on their performance on the test set:

- **First classification stage** (left or right hip): ResNet50 model, 0.89 accuracy
- **Second classification stage** (normal or not normal hip): VGG16 model, 0.98 recall

- **Third classification stage** (operated or not operated hip): VGG16 model, 0.91 recall ● **Fourth classification stage** (arthroplasty or nailing operation): VGG16 model, 1.00 accuracy ● **Fifth classification stage** (intertrochanteric fracture, subcapital fracture or osteoarthritis): VGG16 model, 0.73 accuracy

  The classification pipeline was built based on conditional statements with the aforementioned models being used sequentially (for reference, see Figure 18). Starting from the first stage, where the left or right side needs to be determined, ResNet50 is being used to obtain the prediction without making a difference regarding the next stages. Saying that, regardless of the first classification output, the data will be then given to the next classification stage, where the state of hip will be determined. If the hip depicted in the image is classified as normal, the pipeline stops and it outputs the predictions of the first and second stage (e.g. left hip). If it is not normal, the pipeline leads to the third classification stage. This stage outputs if the image contains an operated or a non operated hip. If the hip is operated, the fourth classification stage is activated, in order to output if the image contains arthroplasty or nailing. If the hip is not operated, the fifth classification stage is activated. This stage outputs if the image contains an intertrochanteric fracture, a subcapital fracture or osteoarthritis. An output example of the whole pipeline is *"Right Not Normal Operated Nailing"*. Each step of the process ensures that each classification task is handled by the model best suited to that particular decision, optimizing overall performance.

7 CONCLUSIONS

Early and accurate diagnosis is critical for ensuring that patients receive timely and appropriate treatment, which can significantly accelerate their recovery and improve outcomes. Automated diagnostic systems, while still in the early stages of development, hold great potential to enhance the diagnostic process. Although such systems should assist, not replace, the expertise of medical professionals, they can help streamline workflows, reduce diagnostic delays, and even minimize the risk of human error in clinical settings.

The work presented in this thesis demonstrates the effectiveness of deep learning techniques, particularly through the application of transfer learning, in automating the diagnosis of hip-related conditions. The proposed multi-stage classification pipeline showcases the capability of these methods to make highly accurate predictions across a range of medical conditions, validating their role as powerful tools in the field of medical image analysis. Transfer learning, in particular, proved crucial in improving model performance, enabling the models to generalize effectively despite the limitations posed by the available data.

However, certain challenges remain, including the limited availability of open-source medical imaging data and the constraints of computational resources for training robust models. Addressing these issues will require concerted efforts, such as leveraging generative models to create synthetic medical images. By expanding the dataset through generative techniques, it becomes possible to further enhance model performance and generalization.

Looking ahead, there is substantial room for growth in this field. With sufficient resources, many of the current limitations can be overcome, allowing the development of highly advanced diagnostic models. Future work can focus on refining the integration of multiple models into unified diagnostic pipelines, as demonstrated in this thesis, not only improving accuracy and efficiency but also offering more holistic

insights into a patient's condition. Such systems have the potential to indicate other areas of concern that may have been overlooked, providing a more thorough analysis of the medical image.

In conclusion, the future of automated diagnosis shows great potential, and with continued advancements, these systems can become significantly beneficial tools in medical practice, helping to ensure timely, precise, and comprehensive patient care.

Bibliography – References – Online sources

1. *Ali, L., Alnajjar, F., Jassmi, H. A., Gochoo, M., Khan, W., & Serhani, M. A. (2021, 3). Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures. Sensors, 21, 1688. 10.3390/s21051688*
2. *Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine learning from theory to algorithms: an overview. Journal of physics: conference series, 1142. 10.1088/1742-6596/1142/1/012012*
3. *Attum, B., & Pilson, H. (2024). Intertrochanteric Femur Fracture. https://www.ncbi.nlm.nih.gov/books/NBK493161/*
4. *Brunner, L. C., Eshilian-Oates, L., & Kuo, T. Y. (2003). Hip Fractures in Adults. https://www.aafp.org/pubs/afp/issues/2003/0201/p537.html*
5. *Callaghan, J. J., Liu, S. S., & Haidukewych, G. J. (2012). Subcapital fractures: a changing paradigm. The Journal of Bone & Joint Surgery British Volume, 94, 19-21. 10.1302/0301-620X.94B11.30617*
6. *Cano, J. (n.d.). VGG-16 neural network architecture. ResearchGate. Retrieved September 4, 2024, from https://www.researchgate.net/figure/VGG-16-neural-network-architecture_fig1_327070011*
7. *Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement, 20(1), 37-46. https://doi.org/10.1177/001316446002000104*
8. *Dogramadzi, S., Atkins, R. M., & Raabe, D. (2014). A system for anatomical reduction of bone fractures. U.S. Patent Application, 343,583. https://uwe-repository.worktribe.com/output/959239/a-system-for-anatomical-reduction-of-bone-fractures*
9. *Egol, K. A., Walden, T., Gabor, J., & Leucht, P. (2022, July). Hip-preserving surgery for nonunion about the hip. Archives of Orthopaedic and Trauma Surgery, 142(2), 1-7. 10.1007/s00402-021-03820-4*
10. *El Naqa, I., Li, R., & Murphy, M. J. (Eds.). (2015). Machine Learning in Radiation Oncology: Theory and Applications. Springer International Publishing.*
11. *Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. Psychological Bulletin, 76(5), 378-382. https://doi.org/10.1037/h0031619*
12. *Four Factors for Fracture Healing: Treatment of Nonunion and Malunion. (n.d.). Penn Medicine. Retrieved January 24, 2024, from https://www.pennmedicine.org/departments-and-centers/orthopaedic-surgery/about-us/excellence-in-moti on-newsletter/archive/2019-newsletter/treatment-of-nonunion-and-malunion*
13. *Garden, R. S. (1961). Low-angle fixation in fractures of the femoral neck. The Journal of Bone & Joint Surgery British Volume, 43(4), 647-663. https://doi.org/10.1302/0301-620X.43B4.647*
14. *General Data Protection Regulation (GDPR) Compliance Guidelines. (n.d.). General Data Protection Regulation (GDPR) Compliance Guidelines. Retrieved October 7, 2024, from https://gdpr.eu/*
15. *Goldacre, M. J., Roberts, S. E., & Yeates, D. (2002, October 19). Mortality after admission to hospital with fractured neck of femur: database study. https://doi.org/10.1136/bmj.325.7369.868*

16. *Gray, J. L., & Fischer, S. J. (2020, November). Hip Fractures. OrthoInfo. Retrieved January 25, 2024, from https://orthoinfo.aaos.org/en/diseases--conditions/hip-fractures/*

17. *Hara, K., Saito, D., & Shouno, H. (2015, July). Analysis of function of rectified linear unit used in deep learning. In 2015 international joint conference on neural networks (IJCNN). 10.1109/IJCNN.2015.7280578*

18. *He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. CoRR. https://arxiv.org/abs/1512.03385*

19. *Huang, G., Liu, Z., & van der Maaten, L. (2016). Densely Connected Convolutional Networks. CoRR. https://arxiv.org/abs/1608.06993*

20. *Koval, K. J., & Zuckerman, J. D. (2013). Intertrochanteric Fractures. In: Hip Fractures. In Hip Fractures: A Practical Guide to Management. SpringerNew York. https://doi.org/10.1007/978-1-4757-4052-3_6*

21. *Lane, N. E. (2007). Osteoarthritis of the hip. New England Journal of Medicine, 357(14), 1413-1421. 10.1056/NEJMcp071112*

22. *LeCun, Y. (2015). Deep learning (Y. Bengio & G. Hinton, Eds.). nature, (521(7553)), 436-444. https://doi.org/10.1038/nature14539*

23. *Lindsey, R., Lachapelle, A., ..., & Potter, H. (2018). Deep neural network improves fracture detection by clinicians. Proceedings of the National Academy of Sciences (A. Daluiski & S. Chopra, Eds.). 115(45)(11591-11596). https://doi.org/10.1073/pnas.1806905115*

24. *Liu, P., Lu, L., Chen, Y., Huo, T., Xue, M., Wang, H., Fang, Y., Xie, Y., Xie, M., & Ye, Z. (2022, September 06). Artificial intelligence to detect the femoral intertrochanteric fracture: The arrival of the intelligent-medicine era. 10.3389/fbioe.2022.927926*

25. *Malunion and Nonunion Fractures. (n.d.). Penn Medicine. Retrieved January 25, 2024, from https://www.pennmedicine.org/for-patients-and-visitors/find-a-program-or-service/penn-orthoplastic-lim b-salvage-center/complex-fracture-care/malunion-and-nonunion-fractures*

26. *Marsh, J. L., Slongo, T. F., Agel, J., Broderick, J. S., Creevey, W., DeCoster, T. A., & Audigé, L. (2007). Fracture and Dislocation Classification Compendium. Fracture and dislocation classification compendium-2007: Orthopaedic Trauma Association classification, database and outcomes committee., Journal of orthopaedic trauma(21(10)), S1-S6.*

27. *Matsuda, D. K. (2014, February). Arthroscopic Osteosynthesis of Femoral Head Malunion. Arthroscopy Techniques, 3(1), 31-34. https://doi.org/10.1016/j.eats.2013.08.009*

28. *Meinberg, E. G., Agel, J., Roberts, C. S., Karam, M., & Kellam, J. F. (2018, January). Fracture and dislocation classification compendium—2018. Journal of orthopaedic trauma, 32. 10.1097/BOT.0000000000001063*

29. *Melton, III, L. J. (1993). Hip Fractures: A Worldwide Problem Today and Tomorrow. https://www.sciencedirect.com/science/article/abs/pii/8756328293903417*

30. *Mont, M. A., Ragland, P. S., Etienne, G., Seyler, T. M., & Schmalzried, T. P. (2006). Hip resurfacing arthroplasty. J Am Acad Orthop Surg., 14(8), 454-63. 10.5435/00124635-200608000-00003*

31. *Moran, C., Wenn, G., Sikand, M., & Taylor, A. M. (n.d.). Early mortality after hip fracture: is delay before surgery important? 483-489. 10.2106/JBJS.D.01796*

32. *Pauwels, F. (Ed.). (1965). Der Schenkelhalsbruch, ein mechanisches problem. 1-138.*

33. *Preeti, D. S. K., & Dhankar, A. (2017). A review on machine learning techniques. International Journal of Advanced Research in Computer Science, 8(3), 778-882.*

https://d1wqtxts1xzle7.cloudfront.net/100542382/3096-6163-1-SM-libre.pdf?1680361743=&response-content-disposition=inline%3B+filename%3DA_review_on_Machine_Learning_Techniques.pdf&Expires=1719775964&Signature=RQm8J7DF2zE3hKpzYeMAq7l7rgrT8KNYnk16whY2TjRvKjU6

34. Ramponi, D. R., Kaufmann, J., & Drahnak, G. (2018). Hip Fractures. https://doi.org/10.1097/TME.0000000000000180

35. Ribani, R., & Marengoni, M. (2019). A Survey of Transfer Learning for Convolutional Neural Networks. 2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T). 10.1109/SIBGRAPI-T.2019.00010

36. Ribeiro, F. R., Takesian, F. H., Bezerra, L. E. P., Filho, R. B., Júnior, A. C. T., & da Costa, M. P. (2016). Impacted valgus fractures of the proximal humerus. 10.1016/j.rboe.2016.01.004

37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3), 211-252. 10.1007/s11263-015-0816-y

38. Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of research and development, 3(3), 210-229. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5392560

39. Shah, V. (2023, December 12). Subcapital fracture | Radiology Reference Article. Radiopaedia. Retrieved January 22, 2024, from https://radiopaedia.org/articles/subcapital-fracture

40. Sheehan, S. E., Shyu, J. Y., Weaver, M. J., Sodickson, A. D., & Khurana, B. (2015, July 17). Proximal Femoral Fractures: What the Orthopedic Surgeon Wants to Know. https://doi.org/10.1148/rg.2015140301

41. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. https://arxiv.org/abs/1409.1556

42. Sutton, R. S., & Barto, A. G. (1999). Reinforcement learning. Journal of Cognitive Neuroscience, 11(1), 126-134. https://d1wqtxts1xzle7.cloudfront.net/6985553/ivry_rev-libre.pdf?1390849262=&response-content-disposition=inline%3B+filename%3DReinforcement_learning.pdf&Expires=1719779246&Signature=Km9T2jWoD6S0zw6vVXDyZslBINZQxFZx~hgMMk4Y22B62GyGkyS0WhP9OmHUJ7Kuyr7NhM5LH

43. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. CoRR. https://arxiv.org/abs/1409.4842

44. Tannast, M., & Siebenrock, K. A. (2009). CHAPTER 7 - Femoral Osteotomy. 64-72. https://doi.org/10.1016/B978-1-4160-5898-4.00007-0

45. Van Otten, N. (2023, May 26). L1 And L2 Regularization Explained & Practical How To Examples. Spot Intelligence. Retrieved September 22, 2024, from https://spotintelligence.com/2023/05/26/l1-l2-regularization/

46. What is personal data? (n.d.). ICO. Retrieved October 7, 2024, from https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/personal-information-what-is-it/what-is-personal-data/what-is-personal-data/#pd5

47. Whiteson, S. (2010). Adaptive Representations for Reinforcement Learning. Studies in Computational Intelligence. 10.1007/978-3-642-13932-1

48. Yıldırım, C., Muratoğlu, O. G., Turan, K., Ergün, T., Mısır, A., & Aydın, M. (2022). The intra- and interobserver reliability of five commonly used intertrochanteric femur fracture classification systems. Joint diseases and related surgery, (33(1)), 187-192. https://doi.org/10.52312/jdrs.2022.498

49. Yinglin, X. (2020). Chapter Eleven - Correlation and association analyses in microbiome study integrating multiomics in health and disease. Progress in Molecular Biology and Translational Science, 171, 309-391. https://doi.org/10.1016/bs.pmbts.2020.04.003