Universidade de Aveiro
2022

Teresa Costeira Narciso

# Deep Learning para a classificação de imagens de raio-X

# Deep Learning for X-ray image classification

**Universidade de Aveiro**
**2022**

**Teresa Costeira Narciso**

# Deep Learning para a classificação de imagens de raio-X

# Deep Learning for X-ray image classification

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Física, realizada sob a orientação científica do Doutor António de Aguiar e Pestana Morais, Investigador Doutorado de Nível 1, associado do Departamento de Física da Universidade de Aveiro , e do Doutor Felipe Freitas, Investigador Doutorado de Nível 1, associado do Departamento de Física da Universidade de Aveiro.

**o júri / the jury**

| | |
|---|---|
| presidente / president | Professora Doutora Florinda Mendes da Costa<br>Professora Associada C/ Agregação, Universidade de Aveiro |
| arguente / examiner | Doutor Jon Ander Gómez Adrián<br>Professor Associado, Universidade Politécnica de Valência |
| orientador / supervisor | Doutor António de Aguiar e Pestana de Morais<br>Investigador Doutorado (nivel 1), Universidade de Aveiro |

**agradecimentos / acknowledgements**

A realização de um trabalho desta natureza não depende apenas de mim, mas de todas as pessoas que participaram diretamente e indiretamente neste processo de aprendizagem.

Quero agradecer em primeiro lugar aos meus orientadores, o Doutor Felipe Freitas e ao Doutor António Morais por me terem dado a motivação indispensável para começar este trabalho e por me terem transmitido todas as ferramentas e conhecimentos necessários para o seu desenvolvimento.

Em segundo lugar ao Doutor António Onofre por me mostrar a importância que deep learning traz nas diversas áreas que me deixou entusiasta para o futuro que me aguarda.

Aos meus colegas Tiago e Samuel, pela continua troca de ideias fundamentais para um conhecimento mais profundo das várias técnicas.

Ás minhas amigas, Patrícia Azevedo, Eva Sousa e Joana Fernandes pela paciência que demonstraram e pelos momentos de aventura que me proporcionaram.

Ao meu namorado, Gabriel Marques que foi essencial no suporte ao longo deste percurso.

E por fim agradeço aos meus pais, Paulo e Joana Narciso e ao meu querido irmão Daniel, pelos valores que sempre me transmitiram e especialmente por toda a confiança e ajuda para alcançar os meus objetivos.

**Palavras-chave** Aprendizagem profunda, Atelectasia, Cardiomegalia, Derrame pleural, Imagens de raio-X, Nódulo, ResNet34

**Resumo**

Este trabalho visa desenvolver um modelo de aprendizagem profunda para classificar correctamente as patologias atelectasia, cardiomegalia, derrame pleural, nódulo e "nenhuma descoberta" em imagens de raio-X frontal.

Dois modelos diferentes foram criados utilizando a arquitectura ResNet34. O primeiro com a clássica abordagem multi-classe e o segundo usa a junção de classificadores binários para classificar atelectasia vs "sem descoberta", cardiomegalia vs "sem descoberta", derrame pleural vs "sem descoberta" e nódulo vs "sem descoberta".

O primeiro classificador alcançou uma área sob a Curva (AUC) da curva característica operacional do receptor (ROC) de 0,73 para atelectasia, 0,91 para cardiomegalia, 0,82 para derrame pleural, 0,67 para o "sem descoberta" e 0,74 para nódulo. O segundo classificador alcançou um AUC de 0,73 para atelectasia, 0,90 para cardiomegalia, 0,78 para derrame pleural, 0,67 para o "sem descoberta" e 0,74 para nódulo. A abordagem multi-classe revelou melhores resultados em comparação com a junção dos classificadores binários.

Ambos os modelos alcançaram os resultados na literatura para as classes de cardiomegalia e derrame pleural; o nódulo e o "sem descoberta" apresentaram os piores resultados sendo muitas vezes confundidos com outras classes. Isto exigiu uma análise mais aprofundada dos modelos, revelando a dificuldade que o modelo tinha na classe "sem descoberta"; o modelo classificou erradamente as imagens "sem descoberta" com grande confiança. Após uma análise mais aprofundada das imagens desta classe, notou-se a má qualidade destas, revelando casos de possíveis imagens mal identificadas e imagens em que os pulmões estavam desfocados.

**Keywords** Atelectasis, Cardiomegaly, Deep-Learning, Effusion, Nodule, ResNet34, X-ray Images

**Abstract** This work aims to develop a deep learning model to correctly classify common chest pathologies like atelectasis, cardiomegaly, effusion and nodule as well as cases where none of these pathologies are present (labelled "no-findings") in frontal-view chest X-ray images.

Two different models were created using the ResNet34 architecture: one with the classic single-label multi-class approach and the other with joined binary classifiers for atelectasis vs no-findings, cardiomegaly vs no-findings, effusion vs no-findings and nodule vs no-findings.

The first classifier achieved an Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve of 0.73 for atelectasis, 0.91 for cardiomegaly, 0.82 for effusion, 0.67 for the "no-findings" and 0.74 for nodule. The second classifier achieved an AUC of 0.73 for atelectasis, 0.90 for cardiomegaly, 0.78 for effusion, 0.67 for "no-findings" and 0.74 for nodule. The multi-class multi-label approach revealed better results compared with the joined binary classifiers.

Both models achieved state of the art results in the cardiomegaly and effusion classes. The nodule and the "no-findings" classes presented the worst mislabeling results. This called for further analysis of the models revealing the difficulty the model had on the "no-findings" class i.e, the model mislabeled the "no-findings" images with high confidence. After more closely inspecting the images in this class, the poor quality of in the images was noticed, revealing instances of possible mislabeled images and images where the lungs were out of focus.

# Contents

# List of Figures

# List of Tables

# Glossary

| | |
|---|---|
| **ML** | Machine Learning |
| **CNN** | Convolutional Neural Network |
| **AI** | Artificial Intelligence |
| **AUC** | Area Under the Curve |
| **DL** | Deep Learning |
| **SL** | Supervised Learning |
| **CT** | Computed Tomography |
| **AP** | Anterior-Posterior view |
| **PA** | Posterior-Anterior view |
| **ROC** | Receiver Operating Characteristic |
| **HLEG** | High-level Expert Group |
| **UL** | Unsupervised Learning |
| **SSL** | Semi-Supervised Learning |
| **RL** | Reinforcement Learning |
| **NN** | Neural Network |
| **ReLU** | Rectified Linear Activation Function |
| **ANN** | Artificial Neural Network |

CHAPTER 1

# Introduction

## 1.1 Motivation and Goals

Medical images are used to diagnose various pathologies in the airways, bones, and cardiac silhouettes without having to operate on the patient. As X-ray images are obtained by non-evasive, relatively safe, and inexpensive methods, they are the most frequent medical image requested by physicians. Interpreting X-ray images requires an experienced radiologist, which is not always possible, especially in developing countries [1].

The development and improvement of Machine Learning (ML) techniques like the Convolutional Neural Network (CNN) have instigated the machine learning community into building models that could help healthcare professionals in diagnosis. If there was a time when good Artificial Intelligence (AI) classifier models were more demanding was during the outbreak of SARS-COV2 (Covid-19). The shortage of medical staff to meet the required needs was predominating, including radiologists. Besides being detected in chest X-ray images, it could also develop pneumonia. As soon as X-ray images with the presence of Covid-19 were available, researchers worldwide started to build AI classifier models to help radiologists with the huge demand. Despite the joint efforts into developing these techniques, AI algorithms struggle to leave the papers into real-world applications. However, results like the ones in [2], where an AUC of 0.96 for Covid-19, showed a promising future for the use of ML as diagnosis tools.

The main drawback with the deployment of ML algorithms for chest X-rays has been the lack of availability of good quality public images datasets as ML algorithms require vast amounts of data for training. The appearance of the ChestX-ray14 database in 2017 [3], which contains 112 120 frontal view chest X-ray images, was one of the first steps to mitigating this issue. This database contains 14 common thorax pathologies. Although the symptoms for these pathologies overlap, like cough and chest pain, in the X-ray images, they have different appearances, which help AI algorithms in the classification task. The attempts in classifying pathologies present in this chest X-ray dataset have been many, with some algorithms even exceeding radiologist performance like the CheXNet in the classification of pneumonia [4] proving how AI can help and exceed the medical professionals.

The work proposed in this dissertation tries to join previous efforts in using AI for classifying pathologies in X-ray images. As the problem involves people's health and life, having a classifier that is accurate in a few diseases is better than having a classifier that detects many diseases but has poor accuracy. For this reason, it was chosen to treat the problem as a multi-class single-label problem. The proposed method uses FastAI to develop two Deep Learning (DL) models to classify in X-ray images the classes: atelectasis, cardiomegaly, effusion, nodule and normal image labelled as "no-findings".

## 1.2 Outline

This document is organised into five chapters. The first chapter (Introduction) presents the motivation and goals. The second chapter (Common chest pathologies and how they are seen in X-rays images) is dedicated to a brief description of atelectasis, cardiomegaly, effusion and nodule pathologies and how they appear in X-ray images with a brief literature review of DL algorithms in this pathologies. The third chapter (Deep Learning) presents the basic theory involved in Supervised Learning (SL), culminating in the description of the ResNet34 architecture. Chapter 4 (Methods, Results and

Discussion) describes the steps to build the classifier models and their respective results. The last chapter (Conclusion) presents the final considerations.

CHAPTER 2

# Common chest pathologies and how they are seen in X-rays images

*This chapter starts with an introduction to how atelectasis, effusion, cardiomegaly and nodule manifests and shows on X-ray images. Then it mentions how ML techniques are being used to create models to help medical professionals at localising evidences of such pathologies in these images.*

## 2.1 Respiratory system and common chest pathologies

The respiratory and circulatory systems provide the organs with oxygen and carbon dioxide expulsion. Figure 2.1 illustrates the essential elements involved in the respiratory system: trachea, lungs and bronchus with the addition of the heart. The bronchioles and alveolus are located inside the lungs. The pleura is composed of two thin membranes that surround and protect the lungs, the pleural fluid, confined within the pleura, enables a smooth sliding of the layers when breathing. The respiratory system is responsible for gas exchange, and the circulatory system transports oxygen and carbon dioxide throughout the body. When we breathe, the air enters and passes through the trachea. At the end of the trachea, the air passes through two small passages known as bronchus. Following the bronchus, the air moves via the bronchioles into the alveolus. It is in the alveolus where oxygen is transferred to the blood. After the oxygen transfer, the heart is responsible for pumping it to all the cells and organs in the body. The body cells consume oxygen and produce carbon dioxide. This carbon dioxide is transported to the lungs where it is diffused to the alveolus and expelled from the body [5].

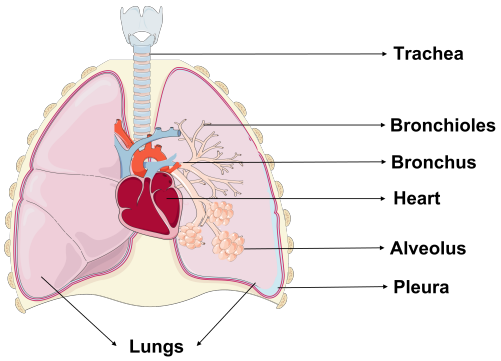



**Figure 2.1:** Identification of some elements involved in the respiratory systems with the addition of the heart. The trachea leads the air to the bronchus, bronchioles and alveolus. The alveolus performs the gas exchange to the blood that the heart pumps to the organs. Revesting the lungs is the pleura, which has a fluid inside. Figure adapted from [6].

Any alterations in the elements in the breathing process can affect health quality. These alterations can have many causes like viruses, infections or diseases, each of which can affect different parts of the respiratory system. Some of the common pathologies in the chest area are:

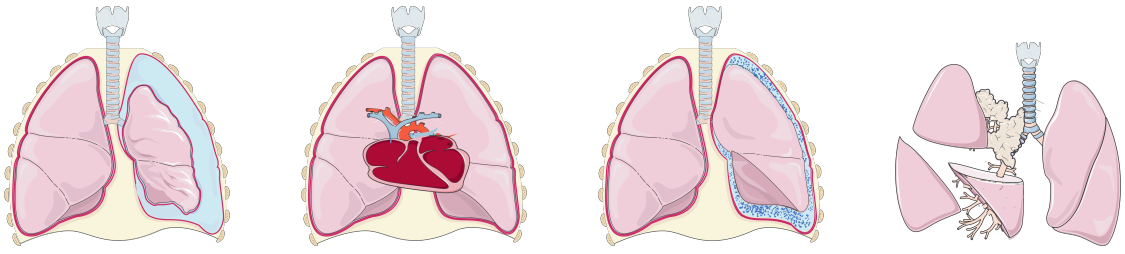
(a) Atelectasis, which manifests as a partial or complete collapse of the lungs, as seen in Figure 2.2a. It may be found in either lung's upper, middle, or lower lobe. General anaesthesia, chest pressure, restricted airways, and other disorders, such as cancer, pneumonia, or pleural effusion, are the most

frequent causes of atelectasis. Common signs of atelectasis include shortness of breath, coughing, chest pain, and increased heart rate; in later stages, cyanosis may appear. It can develop in reaction to increasing surface tension in the alveoli, which causes them to collapse, or as a result of blockage of the bronchi or bronchioles, which causes the lungs to draw inward and the alveoli to lose air. Chest X-rays, Computed Tomography (CT) scans, ultrasounds, or bronchoscopies can all be ordered to diagnose it [7, 8].

(b) Cardiomegaly appears as an enlarged heart, as is illustrated in Figure 2.2b. Cardiomegaly is most frequently caused by obstructions that affect the blood supply, high blood pressure, infections, faulty heart valve, pregnancy, renal illness, drug abuse and HIV; additionally, there might be no apparent cause. It may manifest as a result of cardiac stress, which causes the organ to pump harder and culminate in partial or complete heart enlargement. The typical symptoms are shortness of breath, swollen legs, weight gain, fatigue, palpitations, or missed heartbeats. The diagnosis tools consist of ultrasounds, physical exams, chest X-rays, blood tests, CT scans, MRIs and others [9, 10].

(c) Effusion/pleural effusion consists of the presence of air, pus, blood or other fluids in the space between the pleura. The blue patch in Figure 2.2c represents the extra fluid. The most common effusion causes are pulmonary embolism, cancer, infections, autoimmune diseases, and fluid accumulation from other organs that leak into the pleura. Shortness of breath, chest discomfort, fever, cough, cyanosis, and an increased heart rate are typical symptoms of fluid accumulation in the pleura. It can be diagnosed using ultrasounds, chest X-rays, and CT scans [11, 12].

(d) A Lung Nodule is an abnormal growth of cells in the lungs as Figure 2.2d illustrates. There could be multiple nodules, and they may manifest in one or both lungs. The most common causes for the appearance of nodules include lung-scarring infections, lung cancer, autoimmune disorders and air pollution. In 95% cases, their presence is benign and, in these situations, symptoms are unusual; they only appear as coughing, wheezing, and difficulty breathing if the nodule pushes against an airway. In the malignant kind of nodule, the typical signs consist of chest pain, chronic cough, fatigue, weight loss, recurring respiratory infections and dyspnea. They are usually found using image scans like an X-ray, CT scans, bronchoscopy, and Positron Emission Tomography (PET) [13].



**(a)** Atelectasis **(b)** Cardiomegaly **(c)** Effusion **(d)** Nodule

**Figure 2.2:** Illustration of four common chest pathologies: In (a) is demonstrated a collapsed lung, (b) shows an enlarged heart, (c) fluid or air in the pleura represented by the blue dots, and (d) an appearance of abnormal cell growth. Figure adapted from [6].

## 2.2 Pathologies in the X-ray images

When a doctor wants to examine the patient's interior structures, they frequently request an X-ray image. Projectional radiography (conventional radiography) is often the favoured technique since X-rays can penetrate the body and produce an image that identifies problems in the airways, bones,

cardiac silhouette, and diaphragm [14]. Therefore, a chest X-ray is recommended when patients exhibit symptoms like coughing, trouble breathing, chest discomfort, or fever and their doctor suspects an underlying lung problem.

When an X-ray image is ordered, the patient is brought to the room with the X-ray machine and instructed to remain still between the X-ray source and the X-ray detector, as shown in Figure 2.3. There are portable X-ray equipments where the detector is put under the patient if the patient cannot move. An X-ray beam is produced at the source; this beam passes through the body region of interest for brief seconds, generating an image in the detector [15–18].

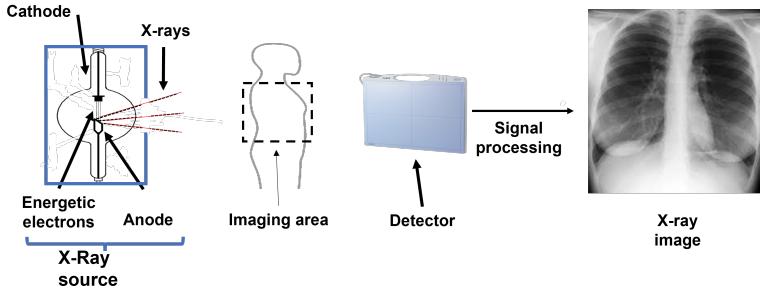



**Figure 2.3:** Schematic illustration of an X-ray machine. The free electrons are generated at the anode, and the application of high voltage allows for them to be accelerated. Their fast movement and collision with the anode produce X-rays. These X-rays escape through an open window in the machine and are directed towards the patient. Depending on the density of the body tissues, some are absorbed, and others are transmitted towards the detector that, with signal processing, returns an X-ray image. Figure adapted from [16].

An X-ray is an electromagnetic wave that carries high-energy photons. When interacting with matter, an X-ray can be absorbed, scattered or transmitted. It consists of ionising radiation, meaning when the X-ray interacts with an atom, it can result in the removal of electrons from the atom resulting in two types of particles: free electrons and an ion [17]. The X-ray machine has two electrodes located in the X-ray source. Through current, a filament known as the cathode is heated. As a result, the thermionic effect manifests and energetic electrons are extracted from the cathode. Free electrons are quickly drawn to the anode (target) by applying a high potential difference. As a consequence, a series of collisions with the target takes place. In these fast movements and collisions, X-rays can be emitted in two forms: bremsstrahlung or characteristic X-rays. In the first scenario, the free electrons decelerate when travelling towards the target; this slow-down releases energetic photons in the form of X-rays. In the second scenario, when a free electron collides with the target if it has an energy equal to the binding energy of the orbital electron in the atom, this orbital electron will be ejected, causing a higher-up electron in the orbital to move to the location of the removed electron to keep the atom stable. The excess energy can be released as an X-ray. Most of the X-rays produced by medical X-ray machines are dominantly bremsstrahlung X-rays (80%), while the rest are characteristic X-rays. Due to the heating, a cold mechanism must be put in place, and both electrodes are positioned inside a vacuum tube to prevent electron collisions with other particles. A leader box with a little opening through which the beam can flow surrounds the vacuum tube. This prevents X-rays from being dispersed. Depending on the atoms in the body, an X-ray beam can either be entirely transmitted or absorbed at various rates. Typically, soft tissue, like the lungs, which contain mostly air, enables photons to pass through, while bones tend to absorb more X-rays due to the denser structures. The remaining X-ray photons that penetrate the body eventually reach the detector, forming a common radiography picture. The detector can either be an X-ray film or an image recording plate. The areas of the detector that receives more X-rays (radiation that passes through the soft tissues) will be

processed as a darker colour. Comparatively, the area where more X-rays were absorbed, which caused the detector to receive less radiation, is processed lighter, like the bones [15–18].

Modern machines enable control over brightness, contrast, radiation dosage, and beam focus. As each radiologist selects the machine parameters based on the body portion being examined and their preferences, the X-ray pictures produced by the radiologists differ. After the procedure, the X-ray image is stored in an electronic file, and the radiologist makes the report and sends it to the physician [15–18].

For the specific case of chest radiography, they can be taken in different positions like frontal, lateral, and rarely oblique. It can be taken from front-to-back (Anterior-Posterior view (AP)), and as back-to-front (Posterior-Anterior view (PA)), depending on where the beam reaches first. The PA view tends to be the preferred method as AP usually is used for supplementary view or if the patient is not able to position himself correctly [19].

Radiologists can distinguish the different pathologies because atelectasis, cardiomegaly, pleural effusion and nodule have different appearances in the X-ray images. Figure 2.4 shows common appearances of these pathologies in frontal X-ray images.

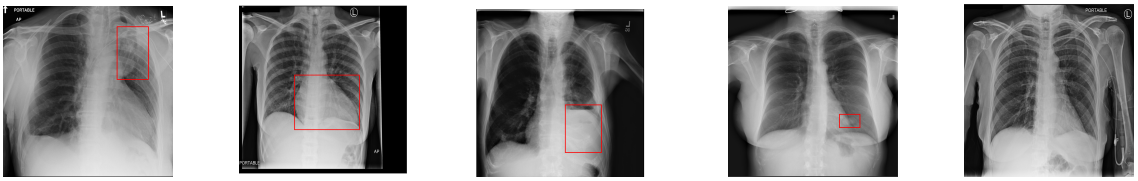
(a) Atelectasis in an X-ray image, the collapsed part of the lung appears either partially or completely white. There could be other signs in the image, including elevation of the diaphragm, narrowing in the space between the ribs, and a shift of the heart and mediastinum to the side where the atelectasis is occurring. Although frontal views are always required, occasionally lateral views tend to be better for detecting atelectasis [20]. An instance of atelectasis can be seen in the left upper lobe (Figure 2.4a), surrounded by the red polygon.

(b) Cardiomegaly can be detected in an X-ray image by measuring the transverse diameter of the heart and dividing it by the diameter of the thoracic cage. It is considered a case of cardiomegaly if this ratio is higher than 0.5. Early signs of heart enlargement can also be detected if there are previous X-rays of the patient. Figure 2.4b shows a typical case of heart enlargement where the heart occupies more than half of the thoracic cage [14].

(c) In pleural effusion, the appearance differs if the patient takes the X-ray scan lying down (supine) or standing (upright). Small quantities of fluid are usually easier to detect from the lateral view than from the frontal view. The fluid is tougher to see in the X-ray image in the case of the supine frontal view because it is located towards the back of the chest cavity. In the upright case, the fluid surrounds the lungs and obscures part of the diaphragm. In the case of Figure 2.4c, the effusion appears at the left lower lobe and appears white, obscuring part of the lung; at the top of the white surface, a concave surface appears [21].

(d) Nodule in X-ray images tends to be identified by the size and rounded opacity. Usually, a pulmonary nodule has a diameter smaller than 30mm, is rounded, homogeneous, has defined borders and is surrounded by normal lung tissue. However, there are other types of nodules called airspace nodules that tend to have 8mm and have irregular margins. Figure 2.4d is a classic appearance of a pulmonary nodule in an X-ray image, which appears as a small white surface with rounded and sharp features [22].

(e) The "no-findings" case is also shown in Figure 2.4e for comparison.



**(a)** Atelectasis **(b)** Cardiomegaly **(c)** Effusion **(d)** Nodule **(e)** No-findings

**Figure 2.4:** X-ray images present in the database [3]. The red polygons indicate the location where the pathology occurs.

## 2.3 Machine Learning in chest X-ray images

In the past decade, researchers have focused on developing machine-learning approaches to enhance the ability of medical professionals and students to identify and locate a pathological X-ray image. This can be confirmed by the appearance of large-scale, publicly accessible X-ray image datasets and a variety of medical image classifiers. However, moving these approaches from paper to real-world practice is challenging mainly due to the models exhibiting disparities in performance, the lack of understanding of AI technology in hospitals and data access issues. The appearance of the NIH dataset opened the way for unrestricted access to X-ray images dataset, allowing the machine learning community to practise on them. Recent deep learning techniques have achieved great success in segmentation techniques like the one in [23]. However, there is still a great struggle due to most of the available databases do not include the lungs mask for the segmentation task, and when it includes, it is only for healthy patients and subtle findings in PA view. As a result, there is still a significant difficulty in deploying machine learning algorithms for X-ray images. The existing databases still have several flaws since radiologists must spend time providing good quality images, labels, and segmentation masks [3, 23–25].

The creators of the Chest-XRay14 (previously known as Chest-XRay8) performed classification tasks in their database (NIH). Theirs consisted of weakly-supervised learning and multi-label multi-class problems with a bounding box generator using four different pre-trained architectures AlexNet, GoogLeNet, VGGNet-16 and ResNet50. Table 2.1 shows their AUC results for the test dataset, which consists of 20% of the total images. Their best result was with ResNet50 using weight loss. No mention of data-augmentation in the correspondent paper [3], was found.

Using a 121-layer DenseNet with a batch size of 32, Laleh Seyyed-Kalantary and Guanxiong Liu [24] compare the bias in deep learning algorithms from 14 diagnostic labels in three public chest X-ray databases: MIMIC-CXR, Chest-XRay8 (NIH) and CheXpert. They conclude that ML algorithms tend to have an unfavourable bias towards females, despite the presence of females being slightly less than males. Moreover, in the Chest-Xray8 database, patients with ages between 60 to 80 years old, also show some unfavourable bias, while the model favoured between 20 to 40 years old patients. The appearance of biases related to sex and race has been observed in classical ML pipelines. Laleh and Guanxiong also showed the use of multiple databases tends to mitigate these biases. They achieved AUC values of the ROC curve, using the test dataset (10% of the total), for the atelectasis, cardiomegaly, effusion, and nodule diseases, which are shown in Table 2.1, using the Chest-XRay8 database [24]. These results are an average of five runs using the same hyperparameters but different seeds. Their data-augmentation consisted of 10-degree random rotation, centre crop and random horizontal flip.

Joseph Cohen et al. in [1] developed a web prototype deep learning multi-class multi-label classifier using the NIH database. Their system was developed with the intuition to show healthcare professionals how ML algorithms work so they could provide feedback. This system is freely available online. Their

interface consists of the actual X-ray image, an image with a heatmap of the regions contributing to the prediction, a list with all labels and the probability of each disease being present. The architecture chosen was the DenseNet-121 trained with the same train/test split as [3]. They compared results by not using and using data-augmentation techniques such as random rotations of 45 degrees, translations with a factor of 15% and 15% scaling. As their best result was for models using data augmentation, only these are presented in Table 2.1, taking the mean of the results for the ten splits they performed on the test dataset.

Andrew Elkins and Felipe Freitas in [25], using a pre-trained DenseNet-121 architecture, trained two different classifiers in the Chest-XRay14 (NIH) database. The first one consisted of a one vs all case, where they classified the disease pneumothorax vs all. In the second model, they built a multi-class multi-label classifier for 14 different diseases. The dataset was split randomly, saving 20% of the images to validate the results. A test dataset was never built. The batch size was set to 64, and data-augmentation was applied with values of random rotation with a maximum of 30 degrees, 50% chance of the image to be zoomed by a 1.3 scale, changing in brightness and contrast ranging between 0 and 0.4. Their AUC in the validation dataset for the pathologies atelectasis, cardiomegaly, effusion and nodule are shown in Table 2.1 represented by [25].

DSouza et al. in [26] used the ResNet34 pre-trained on ImageNet and the stochastic gradient descent optimiser for a multi-label multi-class classifier on the ChestX-ray14 database. The dataset was split into training (70%), validation (10%) and testing datasets (20%) in a way that there is not an overlap of patients. They proved that using variable image sizes for fine-tuning boosts performance as the network builds the model's invariant to the pathology size. Their results for atelectasis, cardiomegaly, effusion, "no-findings", and nodule are shown in Table 2.1 represented by [26].

Bose et al. in [2] used the same database as the previous work with the addition of covid X-ray images. Using the DenseNet-121, image selection techniques and data-augmentation, the authors developed a multi-label multi-class classifier for the 15 labels. Only 13 599 images remained after the authors eliminated low-quality X-ray images. Their results for atelectasis, cardiomegaly, effusion and nodule are shown in Table 2.1 represented by [2].

**Table 2.1:** state of the art results for the four pathologies atelectasis, cardiomegaly, effusion and nodule in the NIH database. Each paper is represented by the corresponding bibliographic reference number.

| Paper | Model | Metrics | Atelectasis | Cardiomegaly | Effusion | No-findings | Nodule |
|---|---|---|---|---|---|---|---|
| [3] | AlexNet | AUC | 0.65 | 0.69 | 0.66 | - | 0.65 |
| | GoogLeNet | AUC | 0.63 | 0.71 | 0.69 | - | 0.56 |
| | VGGNet16 | AUC | 0.63 | 0.71 | 0.65 | - | 0.66 |
| | ResNet50 | AUC | 0.71 | 0.81 | 0.74 | - | 0.72 |
| [25] | DenseNet121 | AUC | 0.76 | 0.91 | 0.83 | - | 0.71 |
| [24] | DenseNet121 | AUC | 0.81 | 0.92 | 0.88 | - | 0.78 |
| [1] | DenseNet121 | AUC | 0.84 | 0.92 | 0.88 | - | 0.79 |
| [26] | ResNet34 | AUC | 0.81 | 0.91 | 0.89 | 0.79 | 0.76 |
| [2] | DenseNet121 | AUC | 0.80 | 0.94 | 0.83 | - | 0.82 |

CHAPTER 3

# Deep Learning

*This chapter offers an easy introduction to theory behind machine learning. It starts by describing the various processes involved in model training, from the loss functions to the metrics used, culminating in the description of the Residual Neural Networks.*

## 3.1 Artificial Intelligence

Nowadays, Artificial Intelligence (AI) can be found practically everywhere. However, the precise meaning of this terminology is a source of different interpretations. Human beings are considered intelligent because they display cognitive, emotional, and social intelligence, analytical and logical reasoning, along with an understanding of mathematical and statistical concepts [27, 28]. So, one can say that AI is the ability of machines (computer devices or networking systems) to mimic human behaviour, such as with big chunks of raw data, be able to interpret the information and decide the best way to solve a problem without the interference of a human [27, 29]. In addition to rational reasoning, experts predict that computers will display emotional and social intelligence [28].

On the report [30], the Joint Research Centre (European Commission) agreed with the previously proposed definition by High-level Expert Group (HLEG) of AI:

*"Artificial Intelligence (AI) systems are software (and possibly also hardware) systems designed by humans that, given a complex goal, act in the physical or digital dimension by perceiving their environment through data acquisition, interpreting the collected structured or unstructured data, reasoning on the knowledge, or processing the information, derived from this data and deciding the best action(s) to take to achieve the given goal. AI systems can either use symbolic rules or learn a numeric model, and they can also adapt their behaviour by analysing how the environment is affected by their previous actions."*

With the appearance of new programming languages, the ability to compile and stream large amounts of data (the internet allowed to connect multiple devices) and especially the capacity of processing and storage of computers, the AI field was able to evolve at a quick pace since the 1990s [29, 31, 32].

In the times following 1990 through now, ML and DL techniques started to appear. This evolution made it possible for an AI system to learn through the brute force of big data and prove its worth in various industries, like banking, marketing and entertainment. It is undeniable the reaches of this field, and it is all achievable because AI can automate repetitive computerise tasks, and recognise patterns in big chunks of data and structures with incredible accuracy through progressive learning [29, 32]. Besides, most of the AI frameworks, datasets, code, and publications are open-source, allowing an acceleration of the field.

## 3.2 Types of Machine Learning Problems

Machine Learning (ML) is a branch of computer science that allows computers to learn and find the best parameters for a model with maximum accuracy. Generally, ML systems work with a large amount of data. When building a ML system, the data is usually provided in three forms: training, validation, and testing data. The training data is used to train the algorithm, the validation is for validating the model while tuning the hyperparameters, and the testing dataset is used to test the

model to unseen data. ML algorithms can be grouped into different categories based on their learning method, incrementally or work type. In the work type case, ML systems can be instance-based or model-based. In instance-based, the ML system compares the new data with previously known data, while in model-based learning, the system detects patterns in the data and creates a predictive model [33]. For incremental learning, ML systems differ in how the data is fed to the system. It could be by using batch learning (offline learning), where the data is accumulated and then trained in batches from time to time or by online learning, where the data is continuously fed to the system [33]. ML algorithms are often categorised according to their learning method. Depending on the amount of supervision during training, algorithms can be Unsupervised Learning (UL), Semi-Supervised Learning (SSL), Reinforcement Learning (RL) and Supervised Learning (SL). The following subsections seek to give a superficial knowledge of these types of learning with particular attention to Supervised Learning.

### 3.2.1 Supervised Learning (SL)

In Supervised Learning, both the training and validation/testing data are labelled. During training, the algorithm tries to find correlations between the data with the same label to create a model that can make predictions on new unlabeled data. Classification and regression problems are common types of tasks that use SL; the primary distinction between classification and regression is that the codomain in regression is continuous while in classification is discrete [34]. Figure 3.1 shows an example of a classification problem where the main goal is to be able to classify triangles, hexagons, and squares correctly. The dataset is split into two groups, one for the training and one for the validation. During training, the model searches for a correlation between the same categories, such as squares, triangles, and hexagons. Then it is tested in the same data type to see if the model is making the correct predictions. Testing the model with non-seen data is essential to measure the accuracy.

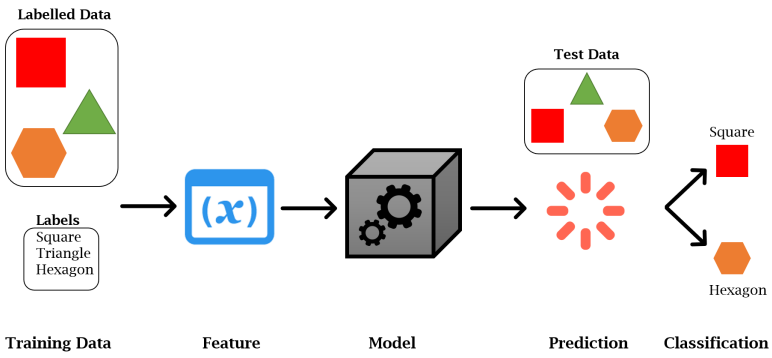



**Figure 3.1:** Basic diagram to explain Supervised Learning.

### 3.2.2 Semi-Supervised Learning (SSL)

In Semi-Supervised Learning, the training data contains more unlabelled data than labelled data. With the partially labelled data, the algorithm creates a model that can correctly label all the unlabelled data. An example of semi-supervised learning is Google Photos. Users provide the person's name in one of the photos; then the algorithm will name them in the rest by recognising the face that was named [33, 35].

### 3.2.3 Reinforcement Learning (RL)

In Reinforcement Learning, the agent (the designated term for the system) lives and observes the environment. The agent responds to its surroundings by selecting and performing actions, leading to distinct rewards, which may or may not be in the form of penalties. The purpose is to develop a policy

that maximises reward over time. Google's alphaGo is an example of a RL; it analysed millions of games before playing numerous games against himself to pick the optimal policy [35].

### 3.2.4 Unsupervised Learning (UL)

The fundamental difference between the Unsupervised Learning and the others is that the data is not labelled. As a result, the algorithm must be capable of extracting some information from the input data [34]. Clustering and dimensionality reduction are two examples of UL. In clustering, the data are grouped depending on their characteristics. Websites are typical examples of clustering, where the algorithm builds various profiles of visitors grouped by age, sex, country or interests. In dimensionality reduction, the algorithm merges co-related features generating a smaller output vector with fewer features than the input. Dimensionality reduction is commonly used with other types of ML as it is suitable for reducing the dimension of the data, permitting it to run much faster without taking up much memory and disk space [33].

This dissertation is an approach to a computer vision problem with binary and multi-classification, which means that the model must be able to classify the images to the correct labels. The following subsections aim to provide a more general understanding of most standard formulations in SL for these cases.

## 3.3 Supervised Problem formulation

The first step in any ML problem is gathering the dataset. The dataset is a collection of data inputs and outputs. These inputs can be images, texts, sounds, numbers, or any other data type, and the outputs are usually numbers, vectors, or strings. The measurable properties of the inputs are known as features [36].

The dataset can be represented as a matrix (Table 3.1). $F = [F_1 ... F_{x_n}]$ is the set of $n$ individual features domains for $m$ data-points. The $x^1 ... x^m$ stands for the data points, while the $y^1 ... y^m$ is the corresponding target label. The dataset can be fed to the system in batches or one by one, depending on the size of the dataset and the performance of the GPU and CPU.

**Table 3.1:** Typical matrix.

| $D$ | $F_1$ | $F_2$ | ... | $F_{x_n}$ | $\mathbf{y}$ |
|---|---|---|---|---|---|
| i=1 | $x_1^1$ | $x_2^1$ | ... | $x_n^1$ | $y^1$ |
| i=2 | $x_1^2$ | $x_2^2$ | ... | $x_n^2$ | $y^2$ |
| ... | ... | ... | ... | ... | ... |
| i=j | $x_1^j$ | $x_2^j$ | ... | $x_n^j$ | $y^j$ |
| ... | ... | ... | ... | ... | ... |
| i=m | $x_1^m$ | $x_2^m$ | ... | $x_n^m$ | $y^m$ |

Assuming that a function that turns $x$ into $y$ exists but is unknown, an approximated function is defined where $(\mathbf{x}, \theta) \mapsto \hat{\mathbf{y}}$. The function that receives the inputs $\mathbf{x}$ and parameters $\theta$ and outputs the predictions $\hat{\mathbf{y}}$ is called a model $h_\theta(x)$. The parameters vector $\theta$ is responsible for fitting the model to the data. The loss function $J(\theta)$ evaluates the model's performance by taking the distance between

the targets $\mathbf{y}$ and the predictions $\hat{\mathbf{y}}$. The loss function returns a low value if the predictions are close to the target and a high value if it is distant, so for a good model, this distance must be as small as possible. $J(\theta)$ can tell how far the model is from the target but does not tell how the parameters must be changed to minimise the loss function. So, the usual way is by using the gradients to estimate if the parameters should be raised or lowered. The training process involves updating the parameters until the model reaches an optimal value [34, 36]. These principles are shown in the diagram of Figure 3.2 where it can be seen the basic steps involved in Supervised Learning such as:

- Loading of the dataset. Organising it by separating the training set and validation set. Assign the labels (targets) to the correct images;
- Initialise the parameters;
- Calculate the predictions using the data in the images and the parameters;
- Evaluate the loss using the predictions and the targets;
- Calculate the gradients (this allows knowing the change needed in each parameter to minimise loss);
- Update the parameters (step) based on the gradient values;
- Repeat the process. Calculate the predictions, loss, gradient, and parameters until the full cycle of steps needs to be stopped.

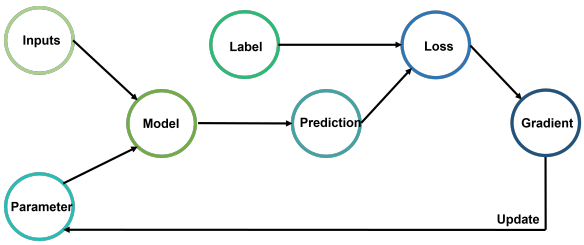



**Figure 3.2:** Basic Loop in a Machine Learning problem.

### 3.3.1 Feature Encoding

The feature domain can be expressed as a string (a.k.a categorical) or as a number. For the majority of ML algorithms, all variables need to be numerical. Integer and one-hot encoding are the two most common approaches for converting string variables into numbers. In both cases, the features domains must be in a discrete form i.e., $F = \{F_1, ..., F_n\}$.

In integer encoding, each categorical feature is assigned to an integer value ranging from $i = [1, 2, \ldots, n]$. In Figure 3.1, where the categories are $F = \{\text{square, triangle, hexagon}\}$, applying integer encoding gives $F = \{1, 2, 3\}$. This form of feature encoding is often used in multi-class problems, meaning each input has only one label.

The one-hot encoding is standard in multi-label problems where the inputs can have multiple labels. In one-hot encoding, the presence or absence of a specific label is represented with a binary vector. Using the example in Figure 3.1, if the feature vector is $F = \{squares, triangles, hexagons\}$ and the first data point (i=1) corresponds to a triangle, then the binary vector for this data is $v = [0, 1, 0]$. For i=2 if the label corresponds to a square, then $v = [1, 0, 0]$, and for i=3 if the label corresponds to a hexagon, then $v = [0, 0, 1]$. Transforming these vectors into a matrix for all the data points, one gets Table 3.2 [34].

**Table 3.2:** Applying the one-hot encoding to the example on Figure 3.1. Where the first datapoint corresponds to the label triangle, the second to the square and the third to the hexagon.

| i | Square | Triangle | Hexagon |
|---|---|---|---|
| i=1 | 0 | 1 | 0 |
| i=2 | 1 | 0 | 0 |
| i=3 | 0 | 0 | 1 |

### 3.3.2 Loss function and activation functions

A loss function measures how well the model is appropriate to the data by using the distance between the target labels, $y = [y^1, ..., y^m]$ and the predicted labels $\hat{y} = [\hat{y}^1, ..., y^m]$. The bigger the loss value more errors the model makes. Different loss functions exist for different ML problems; usually, one can try different loss functions as long as it is appropriate to the data [36].

**Categorical cross-entropy**

The categorical cross-entropy can only be used when the output model is a probability distribution. The cross-entropy is computed according to:

$$CrossEntropy = -\sum_{i=1}^{m} y_i \log(p_i), \tag{3.1}$$

where $m$ is the number of training examples, $y$ is the target label, and $p_i$ is the predicted probability for $y_i$ belonging to the class. The logarithm in the loss function causes wrong predictions to have more impact on the loss function than correct predictions [36].

The activation function SoftMax may be used to convert the model outputs into a probability distribution. In an image, each output of the SoftMax function $\sigma$ gives a probability for each label. So, the sum of the probabilities for each label must have a total of 1. Being $\mathbf{z}$ the input vector and $K$ the number of classes, the SoftMax function can be formulated as

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{3.2}$$

**Binary cross-entropy**

The binary cross-entropy loss is used in binary classification and multi-label classification. The difference between binary and multi-label is that in the first, there are only two labels; in the latter, an image can have $K$ classes, meaning each image can have one or more labels. The binary cross-entropy is defined as:

$$Binarycrossentropy = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{K}\left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right], \tag{3.3}$$

where $p_i$ is the predicted probability for belonging to the class and $1 - p_i$ the probability of not belonging to the class [36].

The logistic function, commonly known as the sigmoid function $\sigma$, is an activation function that converts the Neural Network (NN) outputs into probabilities in binary classification. This activation function can be expressed as

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{3.4}$$

where $z = \theta^T x = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$, $x$ is the feature vector and $\theta$ the parameters vector. This function transforms values much bigger than one into one and values much smaller than zero into zero, meaning the function saturates and is only sensitive for their mid-point [34].

Another standard activation function is the Rectified Linear Activation Function (ReLU), especially in hidden layers on NN. This non-linear function outputs 0 when the input is negative and a positive linear output for the positive's inputs. Although it allows the network to compute efficiently, negative values take the value zero, meaning the gradient takes the value zero and cannot perform back-propagation [34].

**Focal Loss**

The focal loss gives more importance to the samples the model incorrectly predicts than the ones it confidently predicts. Adding a modulating factor to equation 3.1 down-weights the easy samples and allows the model to focus on predicting the complex samples. The focal loss is defined as,

$$FocalLoss = -\sum_{i=1}^{m}(1 - p^i)^\gamma \log(p^i), \tag{3.5}$$

where $\gamma$ is responsible for the down-weighting, the higher the value, the less easy-to-classify samples contribute to the loss [36, 37].

**Label Smoothing**

Label smoothing is an appropriate technique if the model is overconfident, meaning that the probability for the class is extremely high when the model's accuracy is low. The label smoothing is used with the cross-entropy loss or the focal loss. It uses a mixture of the one-hot encoding and adds noise to the data so it can replace the hard 0 and 1 targets with the values:

$$y_{new} = (1 - \beta)y_{one\_hot} + \frac{\beta}{K}, \tag{3.6}$$

where $\beta$ determines the amount of smoothing, $K$ the number of classes, and $y_{one\_hot}$ the previous targets, 0 or 1 depending on the absence/presence for the assigned class [38]. Adding noise to the labels accounts for mistakes that the dataset can contain.

### 3.3.3 Optimiser

The training process involves finding the best values for the parameters vector, $\theta$, that minimises the loss function $J(\theta)$. Finding the new $\theta_{t+1}$ involves computing all the parameters' gradients at the $t^{th}$ time-step. The step size taken during each iteration is known as the learning rate $\alpha$. The gradient value is multiplied by the learning rate, and the product is subtracted from the current value of $\theta_t$ [39]. The process of updating the parameters is defined as

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial}{\partial \theta_{\cdot j}} J(\theta). \tag{3.7}$$

Picking the proper learning rate is crucial; if $\alpha$ is too small, it can take a long time until convergence is achieved; if it is too big, it can pass the minimum and get even bigger values for the loss [36].

There are diverse ways of optimising the neural network. The most common ways are a variation of gradient descent, like the stochastic gradient descent and mini-batch. Gradient descent involves processing the entire training dataset, computing each individual loss, and then averaging the results. The gradient descent only takes the first-order derivatives of the loss function, meaning it only tells the direction and rate to reach the minimum. Although this approach allows for a stable convergence to the minimum, sometimes, it finds a local minimum instead of the global one. It is time-consuming

and requires much computational memory. The stochastic gradient descent is a variation of gradient descent. For this case, the gradient is only computed for one random training example instead of using all the observations. Despite being quicker and requiring less computational effort, this method may produce an imprecise gradient. For the mini-batch gradient descent method, the training set is split randomly into small batches. The number of items in the mini-batch defines the batch size. A different mini-batch from the training set is chosen at each optimisation step. The gradient is calculated the same way as the gradient descent but for the data in the mini-batch. A training epoch is complete once the entire training set has been sampled. This method is a good compromise between the first two allowing for faster training times than the first and better results than the second method [36, 39].

A new gradient descent optimisation algorithm named Adam was presented in 2014 in [40]. This new approach is a variation of known optimiser algorithms like the AdaGrad and RMSProp. Adam reaches the minimum faster and uses less computational power. It calculates adaptive learning rates for each parameter using concepts like momentum and exponentially decaying averages. When one is moving into the minimum location, gradient descent can only tell the direction to follow, but it is not able to tell the type of curvature. The momentum uses the current gradients, $g_t = \nabla_\theta J(\theta_{t-1})_t$, and the past gradients, $g_0 \dots g_{t-1}$, to choose the direction, giving more weight to the latest accumulated gradients. This is known as the exponential weighted moving average, $m_t$, and allows the averaging of the oscillations, which helps to reach the minimum in fewer iterations. Besides the $m_t$, it also computes the squared past gradients $v_t$. If the initialisation for the moments is $m_0 = v_0 = 0$, the updated moments can be computed according to [36, 40]:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t = (1-\beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i \text{ and} \tag{3.8}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 = (1-\beta_2) \sum_{i=0}^{t}, \beta_2^{t-i} g_i, \tag{3.9}$$

where $m_t$ is the estimate of the first moment (mean of the gradient), $v_t$ the second raw moment (uncentered variance of the gradient), $\beta_1$ and $\beta_2$ control the exponential decay rates of the moving averages and $g_t^2$ is the element-wise square $g_t \odot g_t$. As the moving averages are initialised as $m_0 = v_0 = 0$, this results in a bias towards zero. This bias is corrected by dividing equations 3.8 and 3.9 by a factor of $1 - \beta_{1/2}^t$:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \text{ and} \tag{3.10}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \ . \tag{3.11}$$

The final step is to update the parameters using the moving averages according to

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\hat{v}^{\frac{1}{2}} + \epsilon} \ . \tag{3.12}$$

The authors of ADAM [40] propose the values $\beta_1 = 0.99$, $\beta_2 = 0.999$ and for $\epsilon = 10^{-8}$.

### 3.3.4 Metrics

The metrics assess the model's performance in the validation dataset. While the loss function is for the model to assess how well it is doing, the metrics are for the user [36].

This section discusses eight different sorts of metrics where most of which use concepts like true-positive (TP), true-negative (TN), false-positive (FP), or false-negative (FN) [41].

**Accuracy**

Accuracy is the most common metric used in deep-learning algorithms. It gives the ratio of corrected predicted labels in all the dataset. If $\hat{y}_i$ is the predicted value for the $i$-th datapoint and $y_i$ the true value, scikit-learn defines the accuracy for the corrected predictions as [41]:

$$Accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i) = \frac{TP + TN}{FP + FN + TP + TN} \ . \tag{3.13}$$

Using accuracy as a metric can be misleading in a multi-label/multi-class problem with unbalanced data. If one of the classes has a more considerable proportion of the dataset and the model can identify several instances for that class correctly but not the others, the accuracy value will be high even though the model is missing predictions in other classes.

**Precision**

The precision is the ratio of TP in all the positive predictions for a given class. The precision [41], according to scikit-learn, can be defined as equation 3.14 shows. If its value is close to 1, the model is correctly classifying the true positives and is not contaminated with false positives.

$$Precision(P) = \frac{TP}{TP + FP} \ . \tag{3.14}$$

**Recall**

The recall can be computed using equation 3.15. It takes the ratio of the TP for all the samples that should be positive in a given class. The more positive instances the model predicts, the closer the recall is to 1 [41].

$$Recall(R) = \frac{TP}{TP + FN} \ . \tag{3.15}$$

**F-score**

The F-score is a weighted mean of precision and recall. The usual formula to calculate the F-score is defined as:

$$F_\beta = \frac{1 + \beta^2 TP}{(1 + \beta^2)TP + \beta^2 FN + FP} \ , \tag{3.16}$$

where $\beta$ controls how much weight to assign to the recall or precision. If $\beta = 2$, the recall is more valuable than the precision, while if it takes the value $\beta = 0.5$ is the other way around. If $\beta = 1$, both the recall and precision values are equally important. This is known as F1score [41].

**Average precision**

The average precision can only be computed in binary or multi-label classification. It considers the precision-recall curve (PRC) as the weighted mean of precisions at each threshold. According to scikit-learn, if $P_n$ and $R_n$ are the precision and recall at the $n^{th}$ threshold, the average precision is given by [42]:

$$Average\_precision(AP) = \sum_n (R_n - R_{n-1}) P_n \ . \tag{3.17}$$

**Area under the Receiver Operating Characteristic Curve (ROC-AUC score)**

The Receiver Operating Characteristic or ROC curve is a plot that illustrates the performance of a binary classifier at various threshold settings. In a multi-class problem, it must be plotted for one class

vs all the others. The curve plots the true positive rate (also known as recall) against the false positive rate for all the threshold values [41]. The false positive rate is given by

$$FalsePositiveRate = \frac{FP}{TN + FP} \ . \tag{3.18}$$

The AUC in a ROC curve is also essential. The higher the value of AUC, the better the model is.

With most metrics, one can add an average factor to count for unbalanced datasets. This average can be binary, micro, macro, weighted or samples. The binary average only takes into account the results for the positive label. The micro-average calculates the metric globally, meaning that all the TP, FP, FN and FP are independent of the class. The macro-average metric determines each class's score and calculates its arithmetic mean. The weighted average calculates the metric for each class and then accounts for the class's proportion. Finally, in the sample-average, the metric is computed for each instance and uses their average [41].

### 3.3.5 Over-fitting and Under-fitting

Overfitting is when the model fits the training set too well and fails to generalise for the validation and testing datasets. Overfitting might happen when the model memorises the features and trains for excessive time with insufficient data or when the features are too complex and the model focus on the noise. As a result, the model's accuracy worsens, and the training loss decreases while the validation loss increases [36].

Underfitting happens when the features are too complex, and the model fails to create a function that fits these features. During training, the underfit can be seen when the training loss is bigger than the validation loss [36].

The simplest way to avoid overfitting is to find a proper $\alpha$. Low learning rates increase the likelihood of over-fit as the model memorises the features instead of generalising [36].

## 3.4 Deep Neural Networks

Although Machine Learning algorithms enable computers to conduct automatic tasks, it still needs some human intervention. To address this problem, DL employs structured models based on the function of the brain, called Artificial Neural Network. Like a biological neuron, an artificial neuron receives the information, manipulates it and transfers it to the next neuron [34, 43, 44].

Starting with the definition of an artificial neuron. Every time a neuron is mentioned refers to an artificial neuron. A neuron is a single unit mathematical function in an Artificial Neural Network (ANN) which performs a classifier task. Figure 3.3 displays its basic scheme. The inputs are initialised similarly to previously done in section 3.3. However, for an ANN, the parameters vector is formed by the weights, $w$, which determines the influence the inputs have on the output, and the bias, $b$, which is an additional input transmitted to the following layer. The bias prevents the neuron from dying when the input is zero. The inputs ($x_i$) and weights ($w_i$) are multiplied, and the weighted sum of the $i^{th}$ unit is performed according to [34, 43, 44]:

$$S^i = \sum_{i=1}^{m} (\boldsymbol{w}^T)_i \boldsymbol{x}_i + b_i \ . \tag{3.19}$$

This sum is then passed to an activation function, $\sigma$, like the ones reviewed in Section 3.3.2. For a single layer and a single neuron, the output $y_i$, when it is fed with $m$-dimensional inputs $x$ can be written as:

$$y_i = \sigma(\mathbf{w}\mathbf{x} + b) \ . \tag{3.20}$$

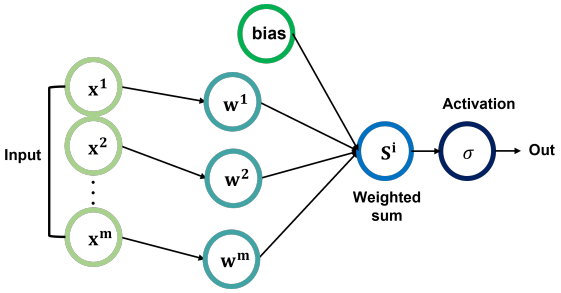



**Figure 3.3:** Basic diagram of a perceptron. The weights are multiplied by the inputs, and the resulting sum is passed to a non-linear activation function. Adapted from [43].

A Neural Network is typically a combination of multiple neurons (nodes) arranged in consecutive layers. Each node in Figure 3.4 corresponds to a neuron. Usually, a NN has three types of layers: an input layer, hidden layers, and an output layer. A simple neural network is one with a single hidden layer, such as Figure 3.3, while a deep layer has multiple hidden layers. These hidden layers perform the transformations of the inputs in equation 3.19. These operations are performed on each layer and passed to the following layers. The more hidden layers the network has, the deeper the NN is and supposedly better the performance due to the addition of more non-linearities [34, 43, 44].

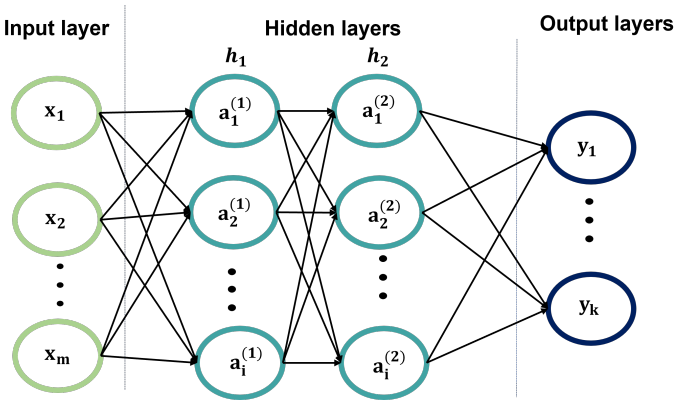



**Figure 3.4:** Basic diagram of a neural network. The input layer contains $m$ examples where $x$ are the inputs. In the hidden layers each node $a_i^j$ is the activation for the $j$ layer and $i$ unit. The output layer contains the outputs for $k$ classes. Adapted from [43].

A fully connected layer (feed-forward neural network) employs a dense matrix of weights for the linear combination, meaning each neuron in the layer connects to all of the neurons in the next layer as in Figure 3.4. There are other layer types, including the convolutional layer and the pooling layer, which are covered in Section 3.4.1 [34, 43, 44].

Training a neural network functions with the same basic steps shown in the initial training loop in Figure 3.2. First, the data is passed through the networks towards the final layers, and the loss function for each unit is computed. Then, the loss function gradient is calculated with respect to all of the weights in each hidden layer using the chain rule, propagating backwards each weight towards the input layer. Finally, each neuron's weights are updated according to the gradients and loop through these steps until training needs to be stopped [34, 43, 44].

### 3.4.1 Convolution Neural Network

A Convolutional Neural Network is a type of Neural Network that revolutionised the computer vision field. When using images as inputs, a multi-connect layer that needs the inputs to be in a vector may eventually lose data in the images. Instead of connecting all the neurons in one layer to all the neurons in the next layer, a Convolutional Neural Network organises each layer into feature maps preserving information about the surrounding neurons according to the weights. A CNN uses a convolutional layer followed by an activation function, a pooling layer and, in the final stage, a fully connected layer.

#### *3.4.1.1 Convolutional Layer*

The convolution layers transform the input tensor into a new tensor. In a computer vision problem, the inputs in a convolution layer are images with three dimensions: height ($h$), width ($w$) and depth ($d$). The first layer receives an input tensor of dimensions $d_i \times n_h \times n_w$, to which a second function known as a kernel with smaller dimensions $d_i \times k_h \times k_w$ sweeps through with a certain value of stride for each of the $d_i$ channels performing elementwise multiplication for the corresponding kernel position. Then the values of the elementwise multiplication for all channels are summed and stored in the output matrix. For the case where $d_i > 1$, and the output channel is $d_o = 1$, each input has its own kernel, and the collection of kernels is known as filters. The results of the elementary multiplication of each kernel with the input tensor are then summed to an output matrix. Figure 3.5, shows an example for two channels ($d_i = 2$) with a stride of 1. The blue colour represents the operations for the first element (56) in the output matrix [34, 44, 45].

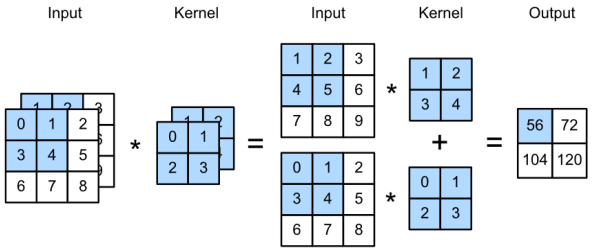



**Figure 3.5:** An example of a filter operation on the input matrix with dimensions of $(d \times h \times w)$. The blue blocks are the values used for computing the first element in the output matrix: $(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56$. Figure from [45].

Usually, in CNN with more layers, the output channels dimensions also increase. For the case of multiple output channels, each output channel $d_0$ has a kernel with dimensions of $(d_i \times k_h \times k_w)$, which means that the convolutional kernel has $(d_0 \times d_i \times k_h \times k_w)$ dimensions [45].

Striding allows the compression of images in a NN. In the striding technique, the kernel shifts along the input matrix with a specific stride value. One can choose the kernel to move $s_h$ pixels for the height and $s_w$ pixels for the width resulting in $[(n_h - k_h + p_h + s_h)/s_h] \times [(n_w - k_w + p_w + s_w)/s_w]$ for the output dimensions [45].

A problem with applying the filter to the images is that the output tensor has dimensions of $(n_h - k_h + 1) \times (n_w - k_w + 1)$ resulting in a loss of pixels at the borders. When one does various convolution layers, this can result in losing important pieces of information. This can be solved by using padding. In the padding technique, an extra pixel is added at the borders of the input images. This translates to more $p_h$ columns and $p_n$ rows in the borders with pixel values of 0. This allows the output shape to be $(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$, with the output shape increasing by $p_h$ and $p_w$ quantities. If the output dimensions must be the same as the inputs, then $p_h = k_h - 1$ and $p_w = k_w - 1$ [45].

At the end of a convolution layer, a non-linear activation function is applied to the output feature map generating an activation map. This activation function is an element-wise operation, so the output has the same shape as the input [45].

#### *3.4.1.2 Pooling Layer*

Following a convolutional layer, a pooling layer exists. Like the convolutional layer, the pooling layer has a fixed shape kernel of $p \times q$ that sweeps the entire input matrix with a specific stride and gives an output for each location. The difference is in the operation that this kernel performs; usually, it outputs the maximum or the mean in that kernel region, meaning there are two types of pooling: max pooling and average pooling. Figure 3.6 shows the outputs for a one-channel input matrix when performing both the max pooling and average pooling. The pooling layer can also use striding, and padding [45].

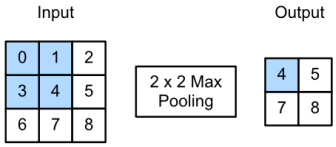



**Figure 3.6:** An example of a filter operation on the input matrix with dimensions of $(d \times h \times w)$. The blue blocks are the values used for computing the first element in the output matrix. Figure from [45].

The final stage in a CNN layer is a fully connected layer like the one shown in Figure 3.4. The outputs of the final pooling layer are flattened for the fully connected layer. This final layer of the network transforms the activation maps provided by the previous layer in a stochastic probability representation of each class.

## 3.5 Residual Neural Network

Although it is often believed that the more layers a NN have, the better its performance, there is a maximum depth a CNN can reach before the accuracy network degrades. This degradation in the NN was hypothesised to be due to the vanishing gradients problem. In deeper networks, the partial derivatives of the loss function start to get smaller and smaller until it gets almost close to zero. The Residual Neural Network emerged in 2015 to address the degradation that existed when deep networks started to converge. This Residual Neural Network introduces the notion of skip connections that skips one or more layers in the NN and adds the outputs layer to the new layer. These added connections permit training deeper models. Using the Residual Neural Network, the gradients flow backwards through the skip connections [46].

The creators of the Residual Neural Network in [46] presented the design for a Residual Neural Network using 34 layers, this network is commonly known as ResNet34, and its architecture is shown in Figure 3.8. Since their publication, new variations of the Residual Neural Network with more layers have been created, and all seem to mitigate the previously known degradation in the CNN. In Figure 3.7, besides showing the architecture for ResNet34 it also shows for the 18, 50 and 101 layers [46].

Considering that Figure 3.7 shows the five blocks for Residual Neural Network compared with Figure 3.8 that shows the design for the ResNet34, each colour box represents a bracketed block; in which the grey is conv1_x , blue the conv2_x, green conv3_x, orange conv4_x and purple conv5_x. The skip connections are represented with full and dot lines between every two convolutional layers. The full line copies the outputs to the new layer, while in the dot line exists a dimension increase. Between these two convolutional layers, there is also a batch normalisation and a ReLu activation [46].

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer |
|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | |
| | | 3×3 max pool, stride 2 | | | |
| conv2_x | 56×56 | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ |

**Figure 3.7:** Architectures for the residual networks with different layers. The brackets represent the blocks [46].

This architecture was trained on the ImageNet dataset, meaning the input size images is $(224 \times 224)$ for the three colour channels. For the block conv1_x, the structure follows [46]:

- a convolutional layer with (3, 64, kernel size $(7 \times 7)$, stride=(2,2), padding=(3,3)), the values 3 and 64 are in and out features to the convolutional layer;
- Batch normalisation;
- ReLu, which outputs $(112 \times 112)$;
- Max pool (kernel size=3, stride=2, padding=1), which outputs $(56 \times 56)$.

The conv2_x has:

- a convolutional layer with (64, 64, kernel size$(3 \times 3)$, stride=(1,1), padding(1,1));
- Batch normalisation;
- ReLu
- a convolutional layer with (64, 64, kernel size $(3 \times 3)$, stride=(1,1), padding(1,1));
- Batch normalisation;
- ReLu
- Repeat all above three times
- Outputs $(28 \times 28)$

At the end of conv2_x there is a dotted line which suggests the size change. The size change can be made by convolutions or padding, but the skip connection is always performed with stride 2. All the other blocks follow the same behaviour with changes in the features [64, 128, 256, 512] and the repetitions' times. For each of the five blocks, the skip connection is at the end of two convolution layers, while the skip connection with the size reduction is at the end of each block. At the end of conv5_x is an average pool layer, followed by a fully connected layer where the output features depend on the number of classes [46].

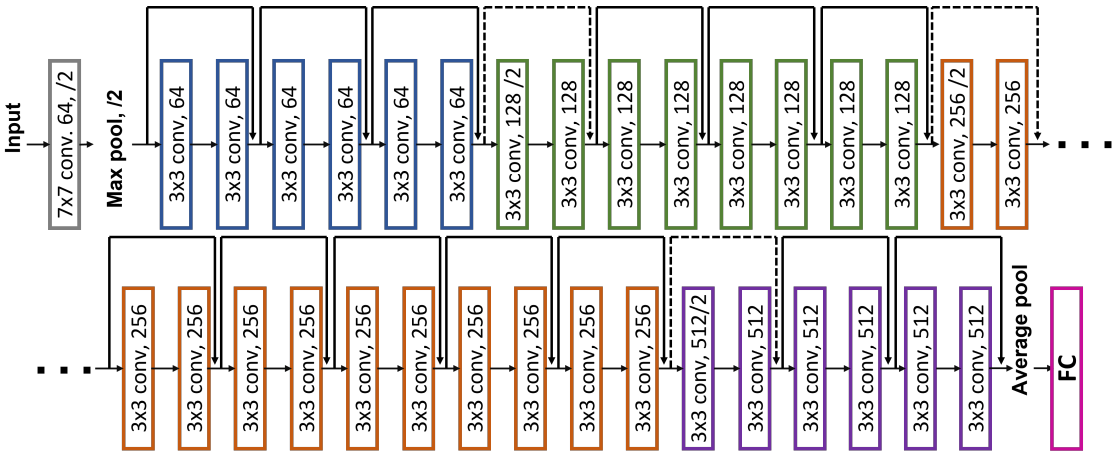



**Figure 3.8:** Residual network for 34 layers. Each colour represents a different block, and the dotted shortcuts the increase in dimensions. Figure adapted from [46].

CHAPTER 4

# Methods, Results and Discussion

*This chapter describes the techniques used for the created machine learning models. A comparison between the multi-class single label for five separate classes and a model that combines four different binary classifiers to categorise the same five classes is performed. This chapter finishes with a final assessment of the suggested methods.*

## 4.1 Dataset

Deep learning-based computer-aided detection of pulmonary diseases mainly relies on public databases of frontal and lateral views for X-ray images and their respective findings. "ChestX-ray8" is a public database consisting of X-ray chest images with eight common pathologies: atelectasis, cardiomegaly, effusion, infiltration, mass, nodule, pneumonia, and pneumothorax. Besides this, it provides images with the no identified pathologies label. Later, the database was upgraded to "ChestX-ray14", which added six more pathologies: consolidation, edema, emphysema, fibrosis, pleural thickening and hernia. The database is accessible online through the website: `https://nihcc.app.box.com/v/ChestXray-NIHCC/folder/36938765345`. It holds a total of 112,120 frontal view X-ray PNG images with a 1024×1024 resolution from 30,805 patients.

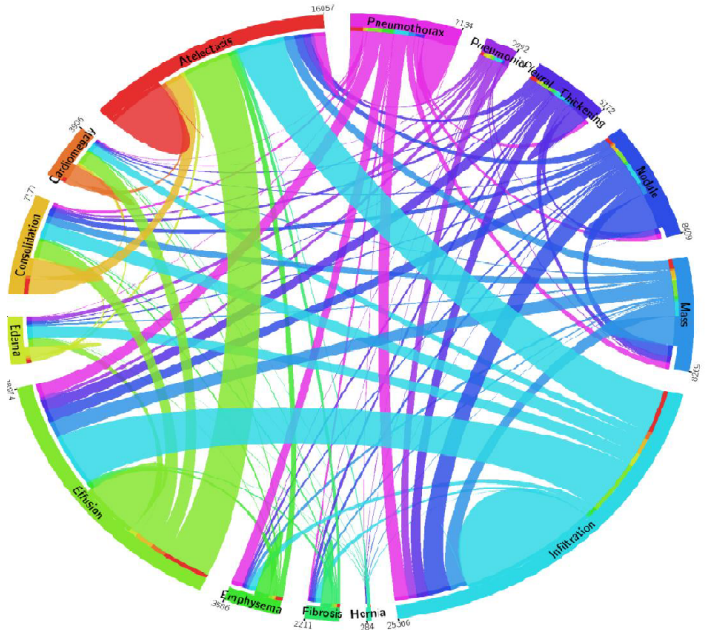



**Figure 4.1:** Distribution and co-occurrence statistics of 14 pathologies in the "ChestX-ray14" database, excluding the "no-findings". Each colour stands for a specific disease. Distribution and co-occurrence statistics of 14 pathologies in the "ChestX-ray14" database, excluding the "no-findings". Each colour represents a certain condition. The lines connecting each illness indicate their co-occurrence. The frequency of each pathology on the dataset correlates with the arc length. Figure adapted from [3]

.

Along with these images, the "Data_Entry_2017_v2020.csv" file also holds metadata for each image, including an image index, finding labels, follow-up, patient ID, age, gender, view position and

finally, original image size and pixel spacing. Through automated extraction techniques, the findings in the X-ray images were gathered from the radiologist reports. Each image has the possibility of multiple diseases, as Figure 4.1 shows [3]. In this graph, each colour represents one pathology, and the lines connecting two diseases mean their co-occurrences. The length of each line in the circumference relates to the number of appearances on the database. For example, infiltration represented with the turquoise colour is the most common pathology on the database, occurs more frequently on its own, and appears relatively equal times with effusion and atelectasis, while fewer times with emphysema and fibrosis.

## 4.2 Fast.AI

The machine learning techniques were implemented using python 3.10.5 programming language with the Fast.AI 2 framework [47]. This framework is built on top of well-known libraries like PyTorch, Numpy, PIL, and pandas and offers users the ability to customise the functions in addition to the ready-to-use tools for training models in computer vision, natural language processing, tabular data, and collaborative filtering domains. The most important classes in the Fast.AI library are the datablock, dataloader and learner.

The datablock is responsible for constructing and gathering the data, following the datablock example below for a multi-class single-label problem using this dataset:

dblock = DataBlock(blocks= (ImageBlock(cls=PILImageBW),CategoryBlock), get_x= ColReader("Image Index", pref= f'path/'), get_y= ColReader("Finding Labels"), splitter = ColSplitter(), item_tfms = Resize(224), batch_tfms= batch_tfm).

This datablock automatically transforms the images to tensors and transforms the label in an integer encoding format, similar to what was done in Section 3.3.1. The get_x and get_y indicate the path for loading the images and the designated target. The splitter is used to create the validation set and training set, which is given by the values in the is_valid column. The item_tfms resizes each image to $224 \times 224$ while batch_tfms, as the name implies, applies batch transformations.

The dataloader is in charge of loading the data when the datablock has been created. The following sequence of commands loads the data in the CSV file where the paths to the images and labels are located; bs denotes the batch size, and the shuffle denotes the datablock to be shuffled.

dloaders = dblock.dataloaders(df, bs=64, shuffle=True, num_workers=8, pin_memory=True).

The Fast.AI's learner class combines the architecture and optimiser and selects the optimum loss function for the user. Another proper function in the Fast.AI library is the vision_learner for transfer learning techniques; it automatically downloads the learned weights in pre-trained models. Bellow, the learner:

learner = vision_learner(dloaders, model, n_out=5, loss_func= loss_func, metrics=[F1Score (average='macro'), accuracy ]).

Pre-trained models make it possible to train more rapidly, accurately and cheap since they can immediately spot patterns in new data, such as edges, gradients, and colour deviations. The CNN learner automatically removes the final layer. This is required since a model's final layer controls the classification for a specific dataset. The last layer is then replaced with one or more layers with randomised initial weights and the correct output size in the last layer, which corresponds to the number of classes in a classification task. A neural network can be thought of as consisting of two parts: the head and the body. The head is responsible for the classification task and typically contains everything after the adaptive average pooling layer. The remaining layers of the model are the body. After the cut of the model head, Fast.AI automatically puts a new head with an average pooling and a

```
(1): Sequential(
  (0): AdaptiveConcatPool2d(
    (ap): AdaptiveAvgPool2d(output_size=1)
    (mp): AdaptiveMaxPool2d(output_size=1)
  )
  (1): Flatten(full=False)
  (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): Dropout(p=0.25, inplace=False)
  (4): Linear(in_features=1024, out_features=512, bias=False)
  (5): ReLU(inplace=True)
  (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): Dropout(p=0.5, inplace=False)
  (8): Linear(in_features=512, out_features=2, bias=False)
```

**Figure 4.2:** Head layers inserted by Fast.AI for the case of a ResNet34 with 2 output classes.

max pooling layer, and then it concatenates the two together (AdaptiveConcatPool 2d layer); finally, it adds two linear layers and dropouts. Figure 4.2 shows an example of the added layers (the head) for the case of a ResNet34 with two output features. While using transfer learning, the model can be trained with the function learner.fine_tune method. The fine-tune method freezes the body and only trains the weights in the newly inserted final layers for a user-specified number of epochs; after the head training is complete, the model is trained for all the layers. So, when calling the fine-tune, it freezes the initial layers and trains the final layer for one epoch, and then it unfreezes all the layers and trains them for the number of epochs requested. The cnn_learner also normalises the image statistics to the statistics that the dataset was pre-trained using batch-normalisation. Another helpful feature of Fast.AI is the discriminative learning rate. This method employs a low learning rate for the first layers because the initial layers do not require a high learning rate as they are responsible for the basics attributes in an image; and a higher rate for the final layers. The fit_one_cycle method is another critical benefit of the Fast.AI library. The one-cycle policy is a sum up of the change in the learning rate and momentum while training. The learning rate is set initially at a low value, progressively increases, and then falls again during the last segment of the training. The momentum (a hyperparameter) fluctuates inversely to the learning rate; it starts with a high value, drops to its lowest and rises again. These two combined reduce the time needed to train a NN and help in stabilisation by preventing the algorithm from falling into steep values of the loss function [36, 48, 49].

## 4.3 Multi-class single-label for 5 classes

The dataset complexity, including the presence of multiple pathologies in a single X-ray image and the class's imbalance, led to the selection of only four pathologies and the supposedly healthy chest X-ray as the variables to be treated. The groups of atelectasis, cardiomegaly, effusion, nodule, and "no-findings" were selected based on how frequently each condition occurs.

The infiltration class is not considered due to being a controversial term in radiographic reports. It is an imprecise and discouraged terminology usually associated with other pathologies that do not help in patient care when used alone [50, 51]. The dataset creators also stated that infiltration is often associated with atelectasis and effusion, so to prevent confusion with these classes, the infiltration was left out [3].

Only the images with a single-label for each one of the five classes were retrieved for it to be treated as a multi-class single-label problem.

### 4.3.1 Searching for the best hyperparameters

Since there is no formula to build a perfect machine-learning algorithm, it is necessary to experiment with some hyperparameters, such as the model's architecture, the best loss function, the optimiser, and the learning rate. In an effort to identify the model that produces the most significant results, many combinations of the various configurations are tested.

This section does not consider the creation of the test dataset, which results in a total of 14 724 images for training and validation: 4215 for atelectasis, 3955 for cardiomegaly, 2756 for effusion, 2705 for "no-findings", and 1093 for nodule. The distribution of the dataset is shown in Figure 4.3. The classes are clearly out of balance, which presents a challenge for neural networks because they often develop biases in favour of the classes that are easier to classify. The function "StratifiedShuffleSplit" of scikit-learn [52] was implemented, saving 20% of the total images for validating the results and ensures the same percentage of classes exists in the validation and training datasets. A new CSV file was created with three significant columns: image index, finding labels and an is_valid column. The first column contains the name of the image formatted as 00001513_000.*png*, the finding labels columns the correspondent pathology found in the image, and the is_valid column contains numbers (1 or 0), 1 if the image corresponds to the training set and 0 if it corresponds to the validation set.

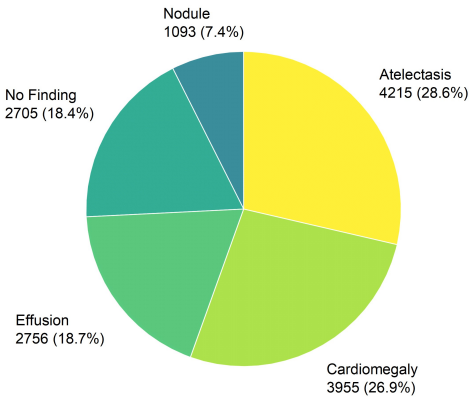
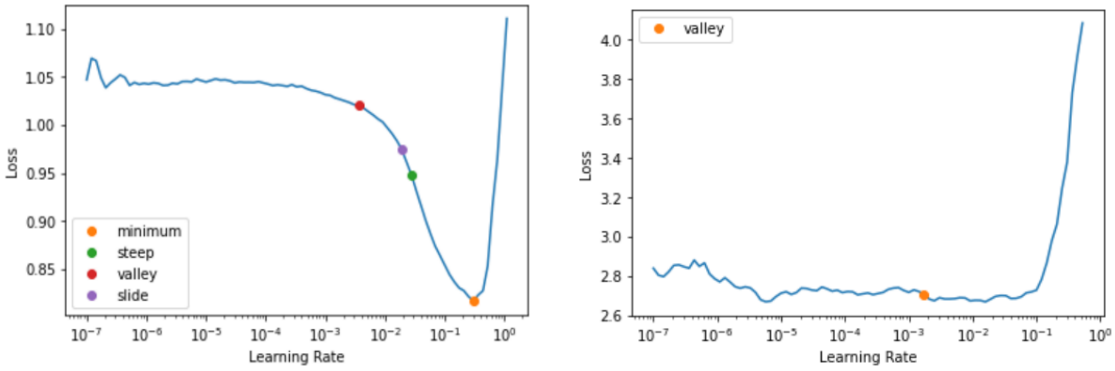



**Figure 4.3:** Distribution of classes in the dataset used for training and validating.

The first thing to do is choose the architecture of the NN to be used. This work only focuses on the ResNet configuration. Only the ResNet18, ResNet34, and ResNet50 designs, which have lower hidden layers, were tested to preserve minimal computational effort. The first layer in a ResNet architecture has the dimensions (batch size, 3, x size, y size), meaning it is built to take three channels from a RGB image. As the images in this dataset are in a grey-scale format, the second thing to do is to convert the images into the RGB format. The next step is choosing a loss function. The networks are dealing with a multi-class single-label problem, the considered loss functions are cross-entropy loss, label smoothing and focal loss. For the optimiser, as Adam shows good results and is the suggested function of the Fast.AI, this optimiser was maintained with the default parameters. Fast.AI uses Adam optimiser with the weight decay approach, and not the L2 regularisation, as the authors in [53] concluded that this approach improves Adam's performance. Nonetheless, when the model was tuned to the best parameters in some final runs, the L2 regularisation was implemented to check if the model had some improvements. Configuration using models trained from scratch and pre-trained weights on the ImageNet dataset is also considered. As the overfit in the models is predominant, each model is trained for 40 epochs using a batch size of 64. Due to the class imbalance, configurations using equation 4.1 as a factor in the loss function are used. This factor attempts to equalise the class weights to reduce its bias. As a result, a predominate class in the dataset is penalised, while a class




**(a)** Expected behaviour. **(b)** True behaviour.

**Figure 4.4:** Plots of the learner.lr_finder.

with fewer occurrences is reinforced.

$$class\_weights_{1,2,\dots} = \frac{n_{samples}}{n_{classes}[n_1, n_2, \dots, n_{classes}]}. \tag{4.1}$$

In equation 4.1, $n_{samples}$ is the total number of images, $n_{classes}$ the total number of classes and $[n_1, n_2, \dots]$ the total number of images for the corresponding class.

Data augmentation is used in some settings to reduce overfitting. Data augmentation is a common technique in machine learning models as it allows a simulation of new images by applying transformations to the existing image. This can be achieved by rotation, cropping, changing the brightness, contrast, applying saturation or inverting the image. In this section, the transformations consist of a combination of different transformations with a maximum rotation of $-3$ to 3 degrees, a maximum zoom of 0.2, and maximum lighting of 0.05 with a probability of 0.3. These transformations are applied to each batch, and every time the batch is called for training, different transformations are performed.

The last important hyperparameter is the learning rate. Fast.AI library offers an easy way to implement the method in [54] using the function learner.lr_find for finding the maximum value of the learning rate in the fit_one_cycle. When using the lr_finder, the expected behaviour is, as Figure 4.4a shows, an accentuated descent until it reaches a minimum and then an accentuated ascent. The problem is that when implementing this function, the model behaves like Figure 4.4b; the recommended approach for this case is to use a value for the learning rate right before the rise. However, the approach followed was to try different values between $\in [3 \times 10^{-6}, 3 \times 10^{-3}]$.

So, recapitulating a total of 101 configurations were tried in search of the best possible model. A combination of the following parameters is tested:

- Architecture: ResNet18, ResNet34, ResNet50;
- Trained from scratch or with transfer learning;
- Loss function: cross entropy, focal loss or label smoothing;
- Adam optimiser with weight decay or L2 regularisation;
- With or without class weights;
- With or without data augmentation;
- Learning rate $\in [3 \times 10^{-6}, 3 \times 10^{-3}]$;

Accuracy, macro-average F1score, and a near absence of overfitting were considered when evaluating how well each model performs.

The initial models provided some early conclusions concerning specific parameter selections, enabling them to be fine-tuned. As a starting point, the range of values for the learning rate was set at

$[3 \times 10^{-5}, 9 \times 10^{-4}]$ since small values took a long time to reach convergence and high values show chaotic behaviour. Models using data augmentation prevented overfitting, so the following models were only run with data augmentation. The last conclusion was that using class weights allowed for an increase in the F1score. In Figure 4.5, the green boxes display an aggregation of the F1scores at the last epoch for all configurations, depending on the architecture. As some configurations were overfitting, these models were removed from assessing the model's performance. The blue boxes in Figure 4.5 are the models that exhibited fewer levels of overfitting. With this figure, some conclusions can be made. The architecture ResNet34 achieves the best results in the blue boxes as 50% of the models achieved an F1score higher than 0.50. The architecture ResNet18 was more prone to overfit; however, the ones that did not were more precise. The ResNet50 architecture had the lowest tendency to overfit, but it covered a wide range of values compared with ResNet34.

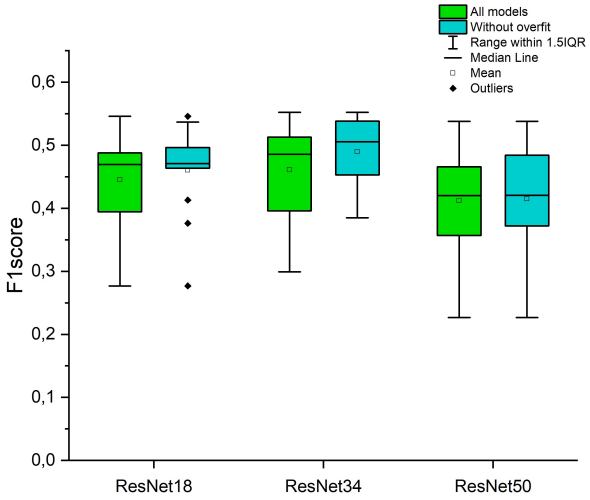



**Figure 4.5:** The box plot displays the distribution results for all trained models depending on the architecture. The F1score metric is chosen to evaluate the best model architecture and parameters. The green boxes show all the models, while the blue boxes are the models that did not display clear signs of overfitting.

Table 4.1 displays the six models that achieve the best F1score. This table shows that four in six have the ResNet34 architecture. The models trained with transfer learning did not reach the top. All use class weights, and four out of six use the cross entropy loss function. Although the accuracy and F1score have low values, this is a good starting point for defining the rest of the work.

**Table 4.1:** The six best configuration models and their F1score and accuracy for the last epoch.

| i | ResNet | Pre-trained | Class Weights | Loss function | Accuracy | F1score |
|---|---|---|---|---|---|---|
| 1 | ResNet34 | False | Yes | CEF | 0.556 | 0.552 |
| 2 | ResNet34 | False | Yes | Label smoothing | 0.555 | 0.550 |
| 3 | ResNet34 | False | Yes | CEF | 0.551 | 0.546 |
| 4 | ResNet18 | False | Yes | CEF | 0.553 | 0.546 |
| 5 | ResNet34 | False | Yes | CEF | 0.549 | 0.541 |
| 6 | ResNet50 | False | No | Focal Loss | 0.548 | 0.538 |

## 4.4 Training the best model for the 5 classes

Deep learning algorithms struggle with medical images due to the image's complexity. The conditions under which the images are obtained do not have default settings, thus, adjustments to the brightness,

contrast, beam focus, and radiation dose of the X-ray machine produce different images. For a chest X-ray, the patient must retain their air and remain still during the beam exposure otherwise, some pathologies may not be correctly seen, and the image could appear blurred. These images also have much information besides the lungs, leading the models to focus on noise and lead to overfitting. This section focuses on exploring the effect when one uses more transformations on the images and increases the values in data augmentation to reduce overfitting and simulate abnormal conditions in X-ray images. This section uses new data-augmentation. The applied transformations in this section are as follows:

- Random rotations between 0 and 65 degrees clockwise or counter-clockwise;
- 50% chance of images to be zoomed by a 1.5 scale;
- 100% chance of an image to be selected to randomly modify brightness and contrast within the range between 0 and 0.9;
- 70% chance of applying warping (translation, shearing, rotation and elastic distortion) by a 0.45 magnitude;
- 50% chance of the image to be vertically flipped;
- Normalise the items according to the stats of each batch, i.e., take the mean and the standard deviation of each channel and normalise the image using them.

These transformations can be seen in Figure 4.6, where eight images from one batch in the testing dataset are present.

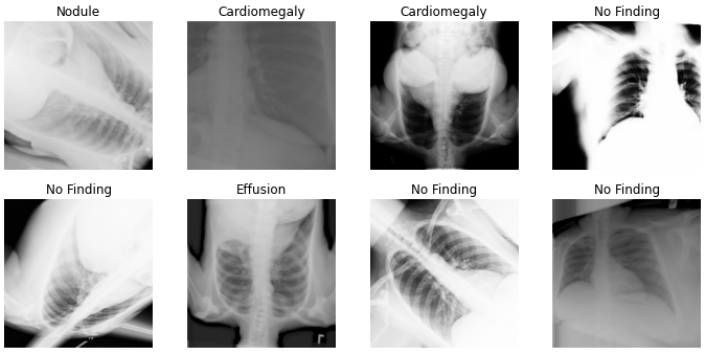



**Figure 4.6:** Eight samples from one batch where the transformations are being used.

All the models discussed in this section use the architecture ResNet34 with class weights in the loss function and a learning rate of $1.85 \times 10^{-4}$ as it is the model with the best results in Table 4.1. Although the cross entropy loss achieves the best results in order to accommodate the uncertainty in the dataset labels, the label smoothing is applied together with the cross entropy loss.

The images used in this section diverge from the last due to the creation of the test dataset. The images included in the test dataset are the same as the file "test_file.txt" from the previously mentioned URL in Section 4.1. The training and validation dataset was created using the same function as the last section, the "StratifiedShuffleSplit" from scikit-learn. The training and validation dataset contains 6285 images of the class "no-findings". The class 'atelectasis' has a total of 3414 images; the 'effusion' has 2788 images; the 'nodule' has 2248 images, and the 'cardiomegaly' has 777 images. The orange columns in Figure 4.7 is the number of images for each class used for training; in green are the ones used for validation, and the purple columns are the images used for testing. Once again, the unbalance in the classes is predominant.

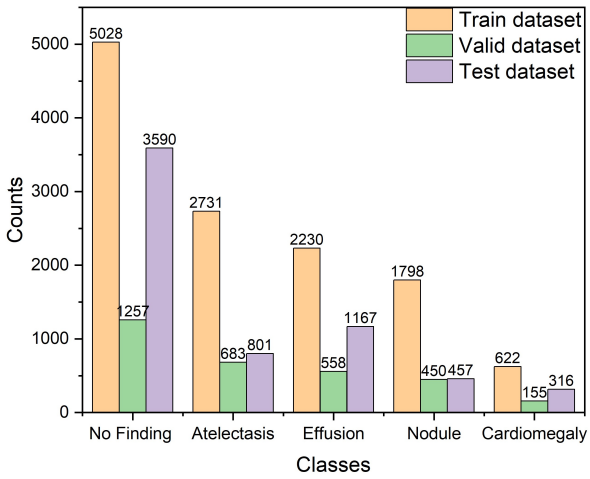



**Figure 4.7:** Class distribution across all datasets. In orange is the training dataset, in green is the validation dataset and in purple is the test dataset. It should be noted that there are differences between the proportions of classes in the testing dataset and the validation dataset.

Table 4.1 showed that all the best models were the ones trained from scratch. The models that used pre-trained weights overfit almost immediately. Although the literature does not agree if pre-trained models are beneficial to medical images, especially X-rays, they are expected to have a faster convergence [55, 56]. Two models were trained to evaluate the statements in the literature: one trained from scratch and the other using pre-trained weights. In Figures 4.8 and 4.9 the losses in the training dataset and validation datasets for both training from scratch and pre-trained models are shown. The symbols represent the saved models with a lower validation loss, highest accuracy and highest F1score.

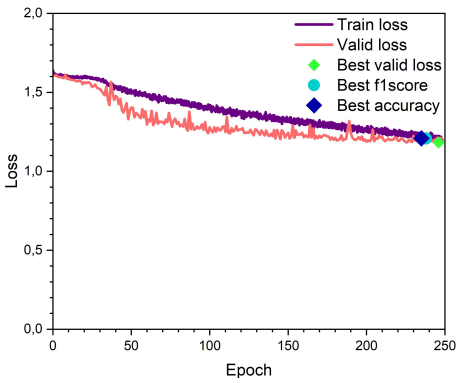



**Figure 4.8:** Train loss and validation loss for the five classes classifier trained from scratch.

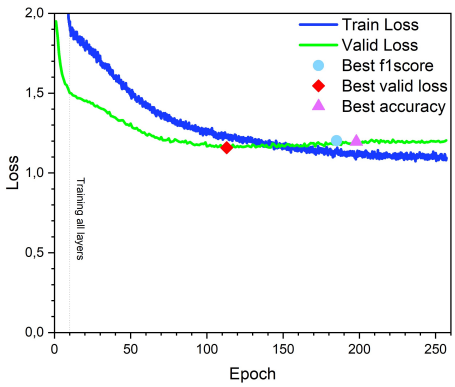



**Figure 4.9:** Train loss and validation loss for the five classes classifier with pre-trained weights.

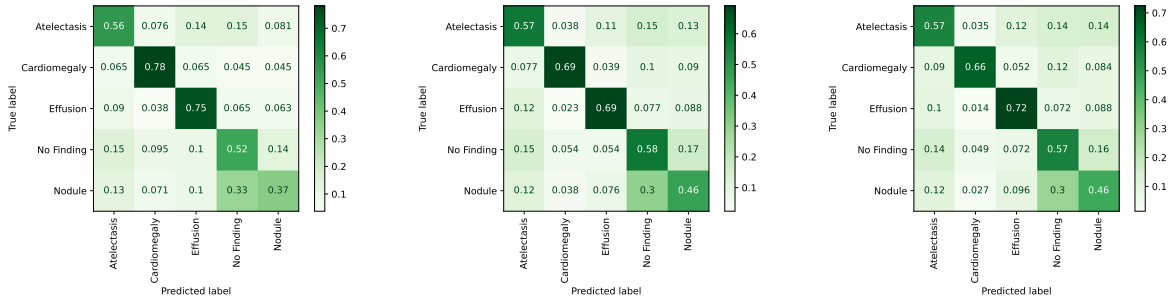
The values of loss, accuracy, and F1score for each saved model in the validation dataset are shown in Table 4.2. The first major conclusion is that training with pre-trained weights achieved better results, with a difference in accuracy and F1score getting to a 3%, achieving these results in early epochs, which confirms the findings in [55, 56]. Second, the use of more harsh data augmentations allows a delay in the overfit. Due to this reason, the discussion of results and the rest of the trained models are done using only pre-trained models.

It is noticed that the best accuracy and F1score for the pre-trained model (Figure 4.9) were obtained for a point where it is overfitting, which means that the metrics are increasing its value when the loss is getting worse. This can be explained by wrong predictions getting worse and thresh-line wrong predictions changing their label to the correct label. The loss function is designed to penalise incorrect

**Table 4.2:** Values of valid loss, accuracy and F1score for each model represented with a symbol in Figures 4.8 and 4.9 with the addition of the metrics in the last epoch.

| Type of learning | Model save | Epoch | Valid Loss | Accuracy(%) | F1score(%) |
|---|---|---|---|---|---|
| Pre-trained | Best valid loss | 114 | 1.159 | 55.9 | 53.4 |
| | Best F1score | 185 | 1.202 | 58.7 | 57.1 |
| | Best accuracy | 198 | 1.195 | 58.8 | 57.0 |
| | Last Epoch | 260 | 1.202 | 58.1 | 56.6 |
| From scratch | Best valid loss | 246 | 1.183 | 52.8 | 52.5 |
| | Best F1score | 238 | 1.207 | 56.5 | 55.0 |
| | Best accuracy | 214 | 1.206 | 57.9 | 54.2 |
| | Last Epoch | 248 | 1.208 | 55.6 | 54.3 |

predictions more strongly, leading to an increase in loss as the model gains confidence in incorrectly classified images. However, the metrics tend to be more resilient and are not affected by incorrect predictions getting worse. Nevertheless, the metrics are impacted by labels close to the threshline shifting the prediction with more training to the correct label. As incorrect predictions impact more strongly the loss function the loss does not improve. The easiest way to confirm is with a confusion matrix and the probabilities for the top losses. The confusion matrix allows us to check if the model is confusing the classes. The lines in the matrix represents the actual classes, while the columns are the predicted classes. In an optimum model, the confusion matrix is expected to be diagonal, which did not happen for either of the models in Figure 4.10. The Figures in (4.10a)(4.10b) and (4.10c) shows all the confusion matrices for the models with the best valid loss, highest accuracy and highest F1score normalise to the true labels for the pre-trained models in Table 4.2.




**(a)** Model saved for best valid loss. **(b)** Model with highest accuracy. **(c)** Model saved for highest F1score.

**Figure 4.10:** Confusion matrices for the models represented with symbols in Figure 4.9 in the validation dataset.

The model is overfitting for the confusion matrices in Figure 4.10b and 4.10c. The classes atelectasis, "no-findings", and nodule are the classes that display improvement compared with the model for the best valid loss (Figure 4.10a), while the classes effusion and cardiomegaly deteriorate. The confusion matrices show that although the model predicts more classes, it loses confidence in the ones it already predicted correctly.

Figure 4.11 displays two mislabelled images in both the valid loss and accuracy models. In both images, the best accuracy model is more confident in the wrong class compared with the best valid loss. It passes from predicting the wrong class with 60% confidence to 65% for the image in 4.11a and 65% for the image in 4.11b. This behaviour suggests that many images might be being predicted with low confidence while getting more confident in the wrong predictions. Therefore, the model used for the rest of the analysis is the one with the lowest validation loss.

As already shown the confusion matrix is not diagonal; the model tends to confuse atelectasis and nodule with the "no-findings" class. The effusion and cardiomegaly images show the best results and fewer signs of confusion. However, the nine higher values of loss are for the cardiomegaly images. This disease tends to be hard on algorithms as the CNN looks for patterns in the image while the radiologists compare the heart diameter ratio with the thoracic cage. Besides these points, the nodule is the more challenging class for the model.

**(a)** "no-findings"/Cardiomegaly/0.6-0.65

**(b)** Effusion/Cardiomegaly/0.6-0.67

**Figure 4.11:** Example of 2 uncorrected labels. The legend in both images must be read as follow: Predicted label/ True label/ Probability for the wrong predicted class in the best valid loss model - probability for the wrong predicted class in the best accuracy model.

#### *4.4.0.1 ROC, AUC and average-precision metrics*

A useful metric to evaluate the model's behaviour is the precision-recall curve and the ROC curve. For an algorithm to be optimum, the area under the curve must be close to one representing a high precision and high recall in the precision-recall curve. As it is a multi-class problem with a high unbalanced of the classes, the precision-recall curve displayed in Figure 4.12 uses the micro-average, meaning that the contribution of each class is averaged over all classes. The average precision obtained a disappointing value of 0.58.

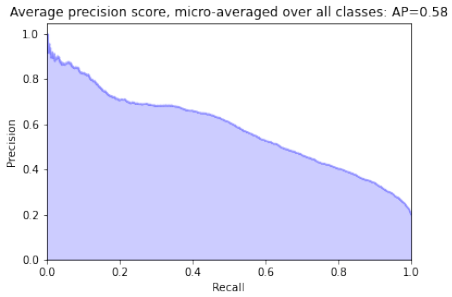



**Figure 4.12:** Average precision-recall for the multi-class classifier using micro-average. The average precision is 0.58 for all classes.

In the ROC curve, a suitable algorithm must be steeper than the straight line representing a random choice, which implies a 50% chance of selecting the correct label. These curves are usually drawn for a binary classifier. Therefore, to compute them for a multi-class problem, they must be considered as a one vs all for each label, demanding casting all the false positives into a list. The full lines on Figure 4.13 show the ROC curves for each class; the green line for cardiomegaly shows the best results with an AUC of 0.94, while the nodule of a pinkish colour as the worst with an AUC of 0.74.

The AUC value for each class is showed on Table 4.3 together with the average-precision for each class. This value reflects the same conclusion as the confusion matrices. The model is having trouble with the "no-findings" and nodule classes. The effusion class seems to have the best results with the

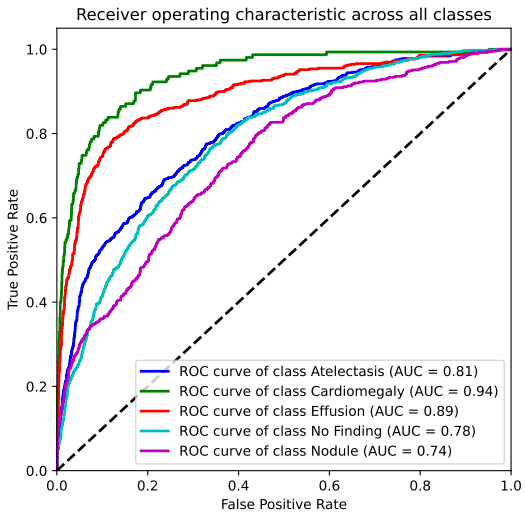



**Figure 4.13:** ROC curve for all the five classes of the multi-class classifier in the best valid loss model in the validation dataset, using the one vs all method to aggregate all the false-positives rates for each class.

higher average precision and second higher AUC. The cardiomegaly, although having a good AUC, has a small average precision.

**Table 4.3:** AUC and average-precision for each class obtained for the best valid model in Figure 4.10a of the 5 multi-class single-label classifier in the validation dataset.

| | **Atelectasis** | **Cardiomegaly** | **Effusion** | **"no-findings"** | **Nodule** |
|---|---|---|---|---|---|
| Average Precision | 0.39 | 0.33 | 0.52 | 0.49 | 0.24 |
| AUC | 0.81 | 0.94 | 0.89 | 0.78 | 0.74 |

Comparing the results in Table 4.3, with the ones obtained in the literature (Table 2.1), the AUC score for atelectasis was only surpassed by the ones in [1]. The cardiomegaly and effusion scores were higher than all of the ones cited, and the nodule scores were only higher than three of the cited articles. However, these results cannot be directly compared as most of the values cited are for a more complex problem.

Given the low results both in accuracy, F1score, AUC, average precision and the confusion matrices in an attempt to improve and help the model reduce the confusion between classes, the following sections cover the creation of binary classifiers for one pathology vs "no-findings".

## 4.5 Binary classifier

The previous results could be better and show a severe difficulty of models with X-ray images as they tend to contain much information. In this section, four binary classifiers are created: atelectasis vs "no-findings", cardiomegaly vs "no-findings", effusion vs "no-findings" and nodule vs "no-findings". These binary classifiers help the models focus on only one disease at a time.

### 4.5.1 Atelectasis vs. "no-findings"

Only the "no-findings" and atelectasis images from the same dataset shown in Figure 4.7 are used. This means that only 35% of the images have atelectasis, so the bias towards the no-finding images

is expected. The data augmentation and optimiser (Adam) stay the same as in the previous section. The loss function is changed to the binary cross-entropy loss with class weights and label smoothing. The architecture uses the previously trained model in the five classes. The previous head layers are removed, and the following new layers are created in order to match the model head to the new output layer:

- AdaptiveConcatPool2d layer;
- Flatten layer;
- Block with BatchNorm 1d, ReLu, BatchNorm 1d, dropout of 0.5 and a linear layer with two out features.

Although the model could have behaved better with the five classes, using the previous model for transfer learning to the new classifiers seams a good choice here once it is expected that the model has learned, in spite of its performance, some essential features of the dataset. Moreover, instead of retraining the model with essentially the same X-ray images, the model can instead focus on learning new features faster to help detect the pathology.

The most significant difference between the previous model and this model is the output. As it is now a binary problem, the output must be either: atelectasis or "no-findings". The head layers are trained for ten epochs using the learner.fine_tune method provided by Fast.AI and all layers were trained by a total of 200 epochs, with a batch size of 64. Figure 4.14 shows the training and validation loss behaviour.

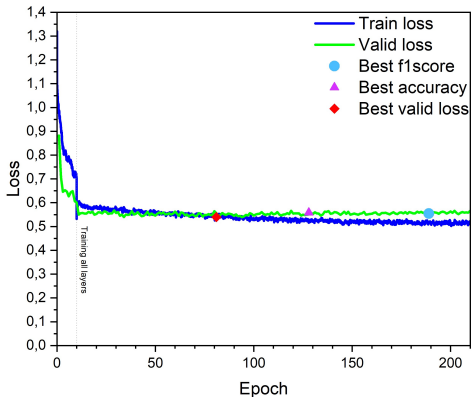



**Figure 4.14:** Training and validation loss for the binary classifier: atelectasis vs "no-findings".

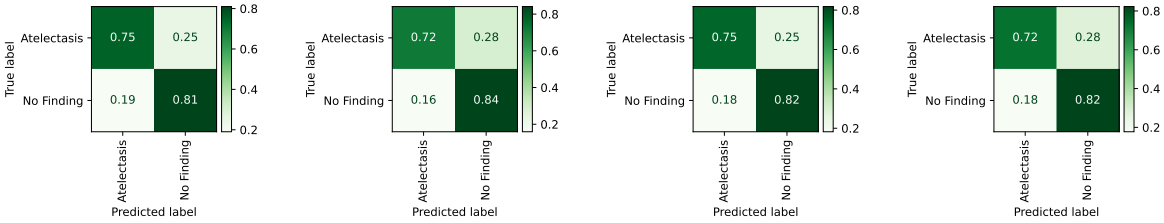
As shown by Figure 4.14, the model starts to overfit around the 100 epoch. The model with the best F1score and best accuracy are expected to have some bias towards the "no-findings" images as they are almost two times more in the dataset and these two models were saved in a region where it is overfitting. Four models were saved: one with the lowest valid loss that achieved a value of 0.5396, one for the best accuracy, one for the best F1score and one for the last epoch. The metrics and valid loss are showed on Table 4.4. The best accuracy achieved a value of almost 80% while the F1score achieved 78%, a good improvement compared with the 59% in accuracy and 57% in F1score in Table 4.2 for the five classifiers. However, this was expected as the dimension of the problem was reduced.

Figure 4.15 shows the normalised confusion matrices for all four models.

Comparing the confusion matrix for the best valid loss (Figure 4.15a) and the confusion matrix for the best F1score (Figure 4.15c), the F1score model achieves better results. However the absolute values shows the model correctly predicts ten more instances for the "no-findings" label in Figure 4.15c, but one less for atelectasis. In table 4.4, the valid loss between the two models has a difference of $-0.0147$ and the F1score of 4.42%. Although the F1score model predicts more correct instances than

**Table 4.4:** Values of valid loss, F1score and accuracy for the four saved models in the atelectasis vs "no-findings" algorithm.

| i | **Saved model** | **Epoch** | **Valid loss** | **Accuracy** (%) | **F1score** (%) |
|---|---|---|---|---|---|
| 1 | Valid loss | 81 | 0.5396 | 78.92 | 73.39 |
| 2 | Accuracy | 128 | 0.5584 | 79.69 | 77.78 |
| 3 | F1score | 189 | 0.5543 | 79.38 | 77.81 |
| 4 | Last epoch | 210 | 0.5661 | 78.81 | 77.04 |

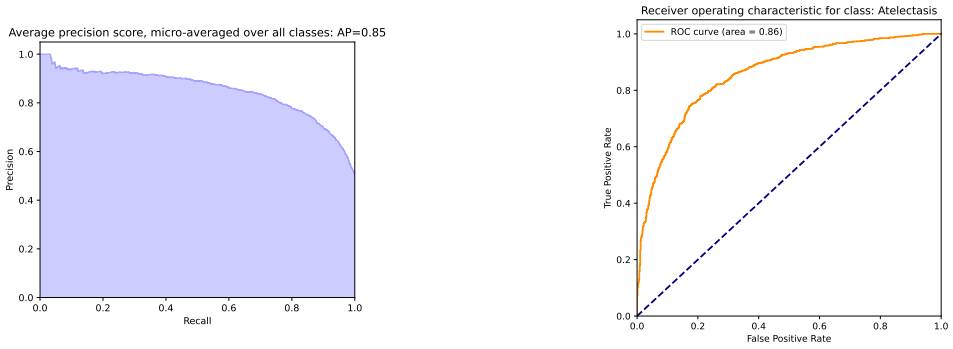



**(a)** Best valid loss. **(b)** Best accuracy. **(c)** Best F1score. **(d)** Last epoch

**Figure 4.15:** Confusion matrices of the binary classifier atelectasis vs "no-findings" for all the models in Table 4.4 in the validation dataset.

the valid loss model, the loss is higher for the first. This is due to the loss function using the class averages to instil the model not to form a bias towards the easiest class (in this case, the "no-findings"), this way the loss is more penalised in the wrong predictions. As the model is starting to show signs of overfitting in the F1score model, wrong predictions are getting worse. Adding to that, this is a classifier for medical conditions, where a prediction of a false healthier condition brings more damages than a prediction of a false disease condition. For this reason the chosen model is the one with the better valid loss.

Joining the discussion with the precision-recall curve and the ROC curve, both showed in Figure 4.16. Both the AUC and the average-precision show an improvement compared with the five multi-class classifier. The AUC increased from 0.81 to 0.86 and the average precision from 0.39 to 0.85. As it has fewer classes to confuse, this was the expected behaviour.




**(a)** Precision-Recall curve with an average precision of 0.85

**(b)** ROC curve with an AUC of 0.86

**Figure 4.16:** Precision-Recall curve and ROC curve for the atelectasis vs "no-findings" classifier in the validation dataset.

### 4.5.2 Cardiomegaly vs. "no-findings"

This section addresses the classifier cardiomegaly vs "no-findings". The process for this class is exactly as the previous one:

- ResNet34 pre-trained on the 5 multi-class classifier;
- Binary-cross-entropy loss with class average and label smoothing
- Head layers trained for 10 epochs and all the layers for 200 epochs
- Four models saved: best valid loss, best accuracy, best F1score and last epoch.

The loss behaviour for both the validation and training datasets is shown in Figure 4.17.

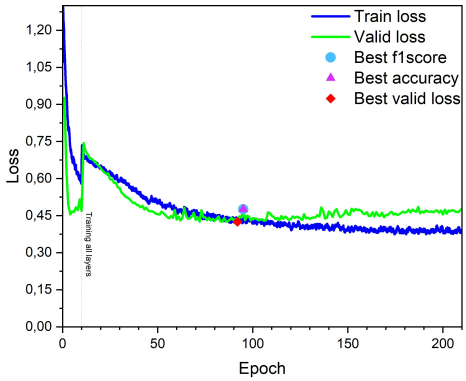



**Figure 4.17:** Training and validation loss for the binary classifier: cardiomegaly vs "no-findings". The symbols indicate the epoch for the occurrence of the best valid loss (rhombus), best accuracy (triangle) and best F1score (circle).

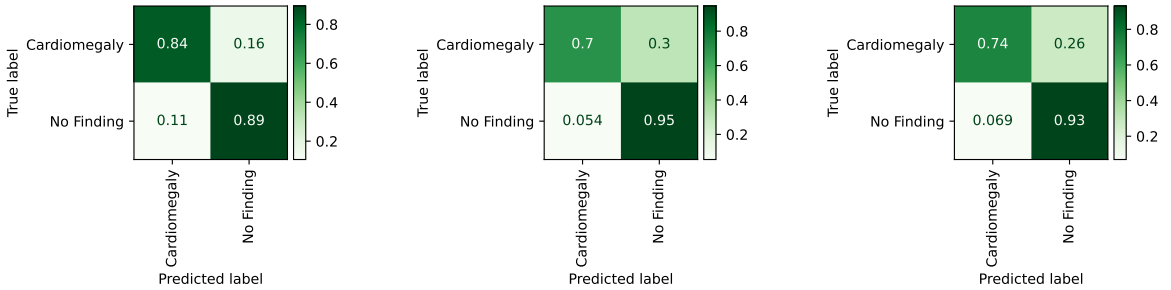
Once again, the model starts to overfit around the 110 epoch. For this case, the best F1score and accuracy occur at the same epoch. Table 4.5 shows the value of the valid loss, accuracy and F1score for all the saved models. Compared with the five classes classifier, the binary classifier shows an improvement in the cardiomegaly class. Both the accuracy and F1score demonstrate better values reaching 90% and 80%, respectively.

**Table 4.5:** Values of valid loss, best F1score and best accuracy for the four saved models in the cardiomegaly vs "no-findings" algorithm.

| i | **Saved model** | **Epoch** | **Valid loss** | **Accuracy (%)** | **F1score (%)** |
|---|---|---|---|---|---|
| 1 | Valid loss | 92 | 0.4250 | 88.81 | 77.82 |
| 2 | Accuracy and F1score | 95 | 0.4757 | 91.86 | 80.32 |
| 3 | Last epoch | 210 | 0.4781 | 90.93 | 79.43 |

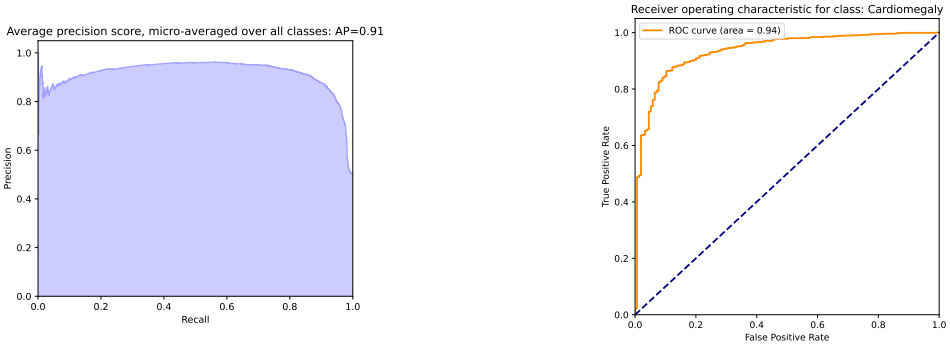
Figure 4.18 shows the three saved confusion matrices. The model saved for the best valid loss (Figure 4.18a) correctly predicts more instances for cardiomegaly than the models for best accuracy and best F1score (Figure 4.18a) and last epoch (Figure 4.18c). The conclusion is the same as for the previous classifier, the best model is the best valid loss model.

The precision-recall curve and ROC curve for this binary classifier is shown in Figure 4.19. The ROC curve is almost perfect, with an AUC value of 0.94 and an average precision score of 0.91. Despite the "no-findings" images composing 89% of the dataset, the model can correctly distinguish the two medical conditions.




**(a)** Best for valid loss. **(b)** Best accuracy and F1score. **(c)** Last epoch

**Figure 4.18:** Confusion matrices of the binary classifier cardiomegaly vs "no-findings" for all the models in Table 4.5 in the validation dataset.




**(a)** Precision-Recall curve with an average precision of 0.91.

**(b)** ROC curve with an AUC of 0.94.

**Figure 4.19:** Precision-Recall curve and ROC curve for the cardiomegaly vs "no-findings" classifier in the validation dataset.

### 4.5.3 Effusion vs. "no-findings"

The effusion vs "no-findings" case was evaluated with the same hyperparameters as before. The effusion class contains 31% of all the images used, so bias towards the "no-findings" is expected. The loss behaviour in the validation and train datasets is displayed in Figure 4.20. The model starts to show minor signs of overfitting around the 130 epoch. The model with the best accuracy and best F1score occurs for the same epoch, so only one is displayed.

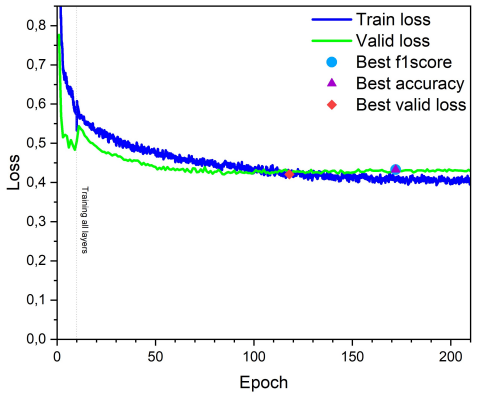
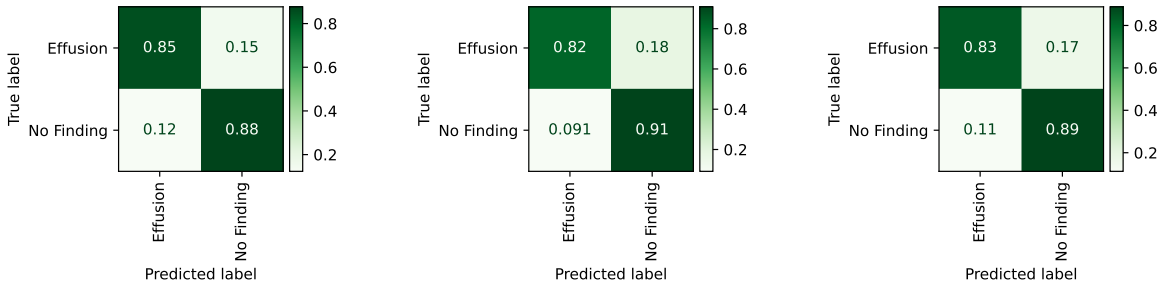



**Figure 4.20:** Training and validation loss for the binary classifier: effusion vs "no-findings". The symbols indicate the epoch for the occurrence of the best valid loss (rhombus), best accuracy (triangle) and best F1score (circle).

Table 4.6 shows the calculated metrics, and loss value for all the models showed in Figure 4.20 and also for the last epoch. The valid loss model displays the worse metrics. However, in the normalised confusion matrices in Figure 4.21, the last epoch model (Figure 4.21c) correctly predicts 83% instances for the effusion class, the accuracy and F1score model correctly predicts 82% (Figure 4.21b), while the valid loss model (Figure 4.21a) predicts 85%, suggesting a deterioration in the class of interest in the last epoch model and best accuracy and F1score model. For this reason, the best model is considered the best valid loss model.

**Table 4.6:** Values of valid loss, best F1score and best accuracy for the four saved models in the effusion vs "no-findings" algorithm.

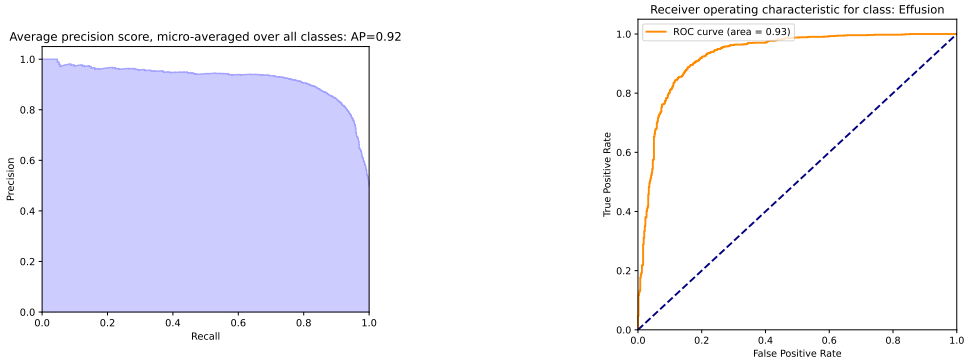
| i | **Saved model** | **Epoch** | **Valid loss** | **Accuracy (%)** | **F1score (%)** |
|---|---|---|---|---|---|
| 1 | Valid loss | 118 | 0.4208 | 86.83 | 85.04 |
| 2 | Accuracy and F1score | 172 | 0.4329 | 88.20 | 86.26 |
| 3 | Last epoch | 210 | 0.4288 | 87.11 | 85.20 |




**(a)** Best valid loss **(b)** Best accuracy and F1score **(c)** Last epoch

**Figure 4.21:** Confusion matrices of the binary classifier effusion vs "no-findings" for all the models in Table 4.6 in the validation dataset.

The plots for the ROC curve and average precision are shown in Figure 4.22. Remembering that this class showed the best results for the five classes classifier, better behaviour in a binary classifier was expected. The AUC score has a value of 0.93 (Figure 4.22b) with an average precision of 0.92 (Figure 4.22a).




**(a)** Precision-Recall curve with an average precision of 0.92.

**(b)** ROC curve with an AUC of 0.93.

**Figure 4.22:** Precision-Recall curve and ROC curve for the effusion vs "no-findings" classifier in the validation dataset.

### 4.5.4 Nodule vs. "no-findings"

The nodule had the worst results in the previous five classes classifier. The procedure here is the same as from the previous classes, with the nodule corresponding to 26% of the total number of images. The validation and training loss is showed in Figure 4.23.

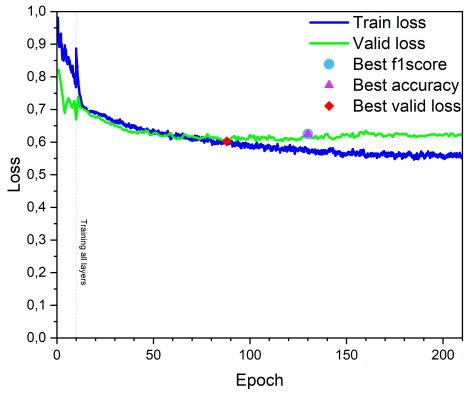
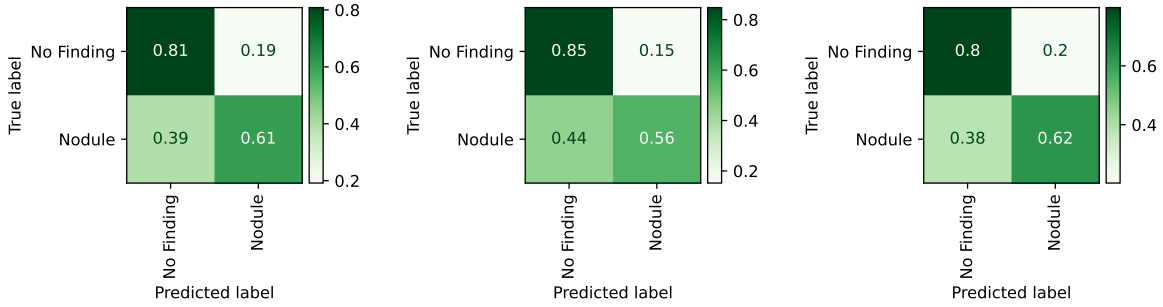



**Figure 4.23:** Training and validation loss for the binary classifier: nodule vs "no-findings". The symbols indicate the epoch for the occurrence of the best valid loss (rhombus), best accuracy (triangle) and best F1score (circle).

The model shows overfitting around the $120^{th}$ epoch. The models with the best valid loss, accuracy, F1score and last epoch were saved. In Table 4.7 the metrics and values of loss for all the saved models are showed. The last epoch model and best valid loss do not show significant difference. The loss has a discrepancy of 0.015, the accuracy 0.29% and the F1score 0.9%.

**Table 4.7:** Values of valid loss, F1score and accuracy for the four saved models in the nodule vs "no-findings" algorithm.

| i | Saved model | Epoch | Valid loss | Accuracy (%) | F1score (%) |
|---|---|---|---|---|---|
| 1 | Valid loss | 88 | 0.6023 | 75.51 | 69.79 |
| 2 | Accuracy and F1score | 130 | 0.6248 | 77.16 | 70.41 |
| 3 | Last epoch | 210 | 0.6174 | 75.22 | 69.82 |

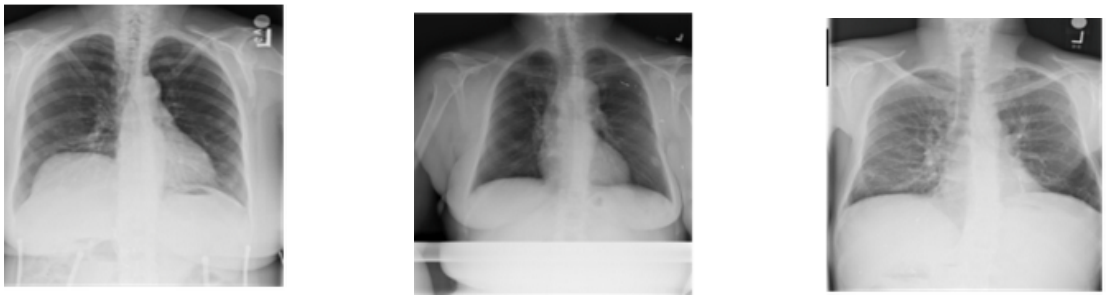



**(a)** Best valid loss **(b)** Best accuracy and F1score **(c)** Last epoch

**Figure 4.24:** Confusion matrices of the binary classifiers nodule vs "no-findings" for all the models in Table 4.7 in the validation dataset

The confusion matrices are showed in Figure 4.24. The structure of the matrices are quite different, when compared with the previous cases. Comparing the last epoch matrix (Figure 4.24c) and the valid loss matrix (Figure 4.24a), the last epoch shows 281 true positives for the nodule class while the valid loss only shows 273. In the last epoch, the class that is being more mislabeled is the no-finding class.

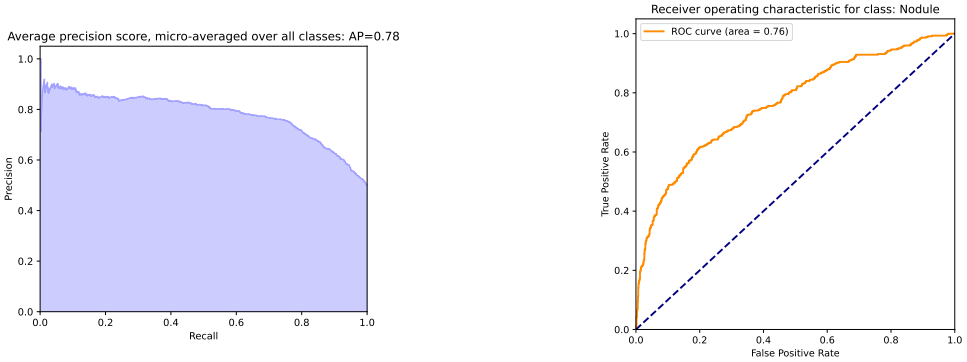
All the nine top losses are images where the actual class is a nodule but the model predicts as "no-findings". The confusion matrices likewise show this behaviour, with the nodule images being classified as "no-findings" between 39 to 44% of the instances. Compared to the valid loss model, the model's confidence in the incorrect predictions is higher in two of the three examples in Figure 4.25.



**(a)** "No-findings"/Nodule/ 0.90-0.84 **(b)** "No-findings"/Nodule/ 0.95-0.92 **(c)** "No-findings"/Nodule/ 0.90-0.92

**Figure 4.25:** Example of three mislabel images. The legend in the three images must be read as follow: Predicted label/ True label/ Probability in the last epoch model - probability in the best valid loss model.

The PR and ROC curve are showed in Figure 4.26. The nodule has the worst results of all the binary classifiers, with an average precision of 0.78 and 0.76 AUC. There was an improvement compared with the multi-class classifier, but it is still worse than the best values in the literature (Table 2.1). Even though employing a binary classifier made it possible to categorise more examples accurately, the algorithm still finds it extremely difficult to recognise nodules in X-ray images.




**(a)** Precision-Recall curve with an average precision of 0.78

**(b)** ROC curve with an AUC of 0.76

**Figure 4.26:** Precision-Recall curve and ROC curve for the nodule vs "no-findings" classifier in the validation dataset.

### 4.5.5 Binary classifier vs. multi-class classifier in the validation dataset

The goal is now to join all the binary classifiers and try to make a five multi-class classifier. Each binary model receives all the images in the dataset of section 4.1. When one passes an image with a class the model was not trained for, and it tries to predict the label for that image, ideally, it should classify as "no-findings". This image prediction comes with two probabilities for each class. So if we pass one image through all four binary models, that image will have a total of eight predictions: four for the "no-findings" and four for all the others. All these predictions are stored in a table with ten primary columns; the first two are the image index and the finding label, and the last eight contain the eight probabilities predictions for each image. The probability for all the "no-findings" is averaged,

resulting in 5 probabilities for each image. The sum of five probabilities must be equal to 1, so the probabilities are normalised for each image. Finally, with the normalised probabilities, the predicted label for each image is the one with a higher value, resulting in the confusion matrix in Figure 4.27.

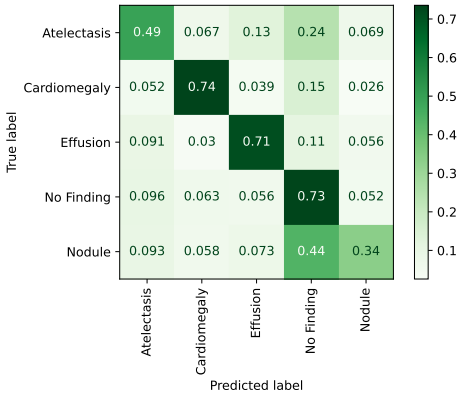



**Figure 4.27:** Confusion matrix for the binary classifier working as a five multi-class classifier in the validation dataset.

Comparing the confusion matrix in Figure 4.27 with the multi-class classifier in Figure 4.10a; the atelectasis images were 7% more mislabelled; moreover, the confusion with the other classes decreased except for the "no-findings" images. The cardiomegaly and effusion images were 4% more mislabelled, and the confusion with "no-findings" images increased. The binary models got 3% less of the images with the nodule right while they confused 11% more often images with nodules with "no-findings" images. The only class with a significant gain in classification —+21%—and an improvement in confusion was the "no-findings" images.

In conclusion, joining the binary models was only proved effective in the "no-findings" images; all the other models were less precise.

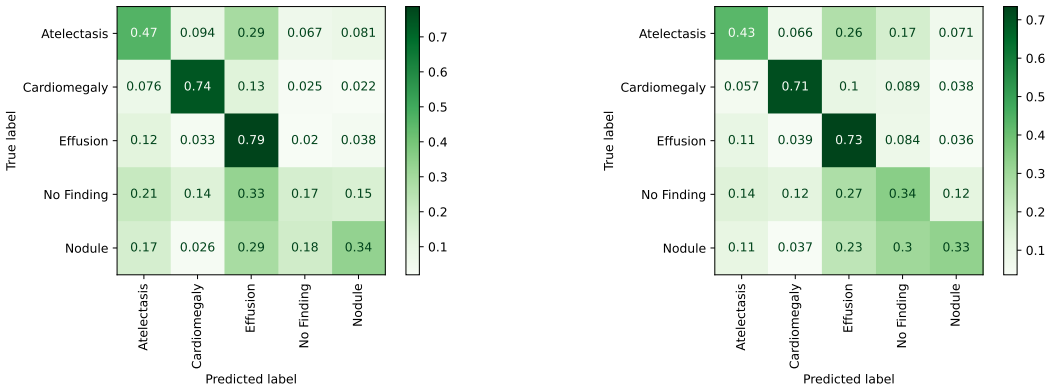
### 4.5.6 Test dataset results

Until now, the test dataset was set aside, and all the results were only for the validation set. As the best possible model has been found, it is time to start predicting the images in the test dataset. This inference uses the best valid models either with the multi-class classifier or the binary classifiers working as a multi-class.

To get the predictions on the test dataset, the first thing to do is to reconstruct the same dataloader, except for the shuffle part, which must be set to false. The model is loaded through the learner.load_model() method. The Fast.AI allows for creating the test dataloader with the function test_dl = learn.dls.test_dl(df_test, with_label=True), in this function df_test, is the pandas' data frame that contains the image name and the finding label for the test dataset. With the test_dl only rests on passing it through the learner.validate(dl=test_dl) and through the learner.get_preds(dl=test_dl,with_decoded=True, reorder=False) to get the predictions and probabilities for each image.

Extra steps are taken to obtain the results for the binary models working as a multi-class. First, the learner.get_preds(dl=test_dl,with_decoded=True, reorder=False) must be passed through all the binary models in Sections 4.5.1, 4.5.2, 4.5.3 and 4.5.4. Then, the same as section 4.5.5 applies. The "no-findings" probabilities are set to the mean of all the predictions for "no-findings" given by the binary classifiers. Then the probabilities for each image are normalised. The predicted label is the one where the probability has a higher value.

Figure 4.28 exhibits the confusion matrix for the test dataset for the multi-class classifier as well as the binary classifiers working together as a multi-class classifier. Although it was anticipated that the predictions in the testing dataset would be worse, the "no-findings" class is given unsatisfactory results having high confusion with all the other classes. However, the binary classifier working as a multi-class which revealed the worst results for all the classes except for "no-findings" images, is mitigating this error.




**(a)** Multi-class confusions matrix

**(b)** Binary working as a multi-class

**Figure 4.28:** Confusion matrix for the test dataset using the best valid model.

Starting by comparing the results of the validation dataset (Figure 4.10a) with the testing dataset in the multi-class classifier, it is worth remarking the following:

- The atelectasis images decreased the true positive instances and increased the confusion with effusion images, and fewer images are being confused with "no-findings".
- The cardiomegaly X-ray images were less accurately predicted, and more images are confused with effusion.
- The effusion images are more often correctly labelled.
- In the no-finding images, there are more instances of false effusion positives than true "no-findings" positives. The model has trouble distinguishing between healthy and pathological X-rays.
- In the nodule class, the confusion grew for every other class; however, for the testing set, the class that the model most confuses is the effusion instead of "no-findings" as was the case in the validation dataset.

Comparing the results of the validation dataset (Figure 4.10a) with the testing dataset in the binary working as a multi-class classifier, the following can be conclued:

- Fewer atelectasis images are correctly labelled, and more of them are classified as effusion;
- The cardiomegaly has fewer images being labelled as "no-findings", but fewer images are being correctly classified;
- The effusion class increased the actual positive instances. The behaviour in terms of confusion with other classes stayed almost equal, but effusion images started to be confused more with atelectasis instead of "no-findings".
- Both the "no-findings" and nodule images got drastically worse.

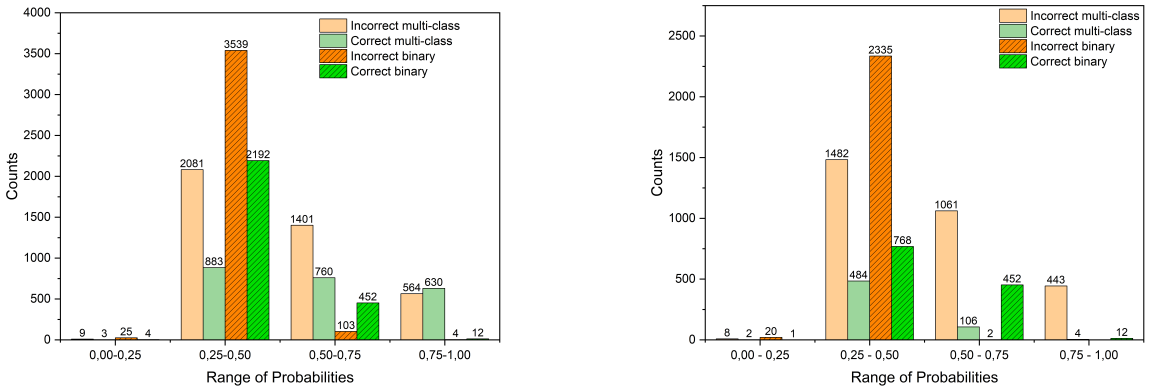
When comparing the two classifiers in the test dataset, the multi-class classifier using binary models classifiers, fewer images are falsely predicted as pathological despite the model performing worse in pathological images. By examining the confusion matrices, it is evident that many images are mislabelled as 'effusion'. When trying to compare the values in the testing dataset for the AUC and

**Table 4.8:** AUC and average precision scores across all classes for both the multi-class and binary model in the test dataset.

| Model | Metrics | Atelectasis | Cardiomegaly | Effusion | "no-findings" | Nodule |
|---|---|---|---|---|---|---|
| Multi-class | Average Precision | 0.20 | 0.24 | 0.35 | 0.57 | 0.11 |
| | AUC | 0.73 | 0.91 | 0.82 | 0.67 | 0.74 |
| Binary | Average Precision | 0.13 | 0.05 | 0.18 | 0.60 | 0.07 |
| | AUC | 0.73 | 0.90 | 0.78 | 0.67 | 0.74 |

average precision with the results achieved in state of the art (Table 2.1), these values are not far off from the ones of the NIH database creators [3].

For a model to be confident in a prediction, the probability at which it assigns the corresponding label must be high. Figure 4.29 shows how many images were predicted with low and high probability for the case of all images (Figure 4.29a) and the "no-findings" images (Figure 4.29b)




**(a)** Predictions in all classes

**(b)** "No-findings" predictions

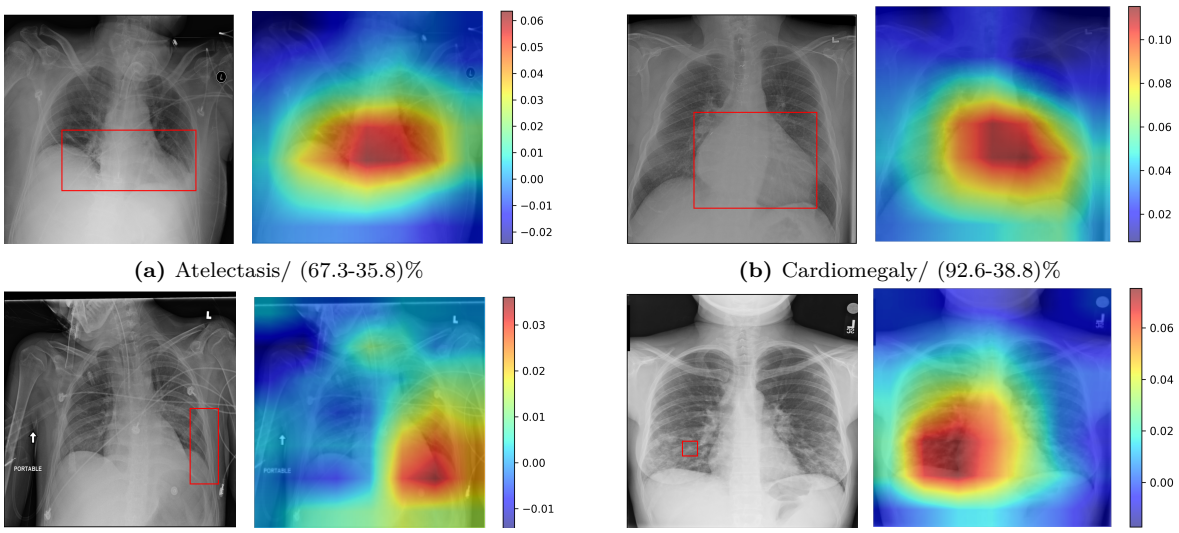
**Figure 4.29:** Aggregated probabilities in which the model assigns the predicted label both for the correct predictions (green) and the incorrect predictions (orange).

Figure 4.29a shows that the model has lower confidence when it makes the incorrect predictions. For the case of the 5 multi-class classifier (pastel colours in Figure 4.29), in 61.1% of the correctly predicted images it makes the correct prediction with more than 50% probability. The five multi-class classifier model has high confidence in the correct prediction and low confidence in the wrong predictions, which shows that even in the instances the model fails, it is not too sure in that prediction. The model that joins the binary classifiers (more vivid colour in Figure 4.29) shows the model has low confidence in the correct predictions and low confidence in the incorrect predictions. Comparing both plots in Figure 4.29, the most incorrect predictions are due to mislabelling in the no-finding images.

For the 5 classes classifier, the model in the "no-findings" images (Figure 4.29b) is making incorrect predictions with more than 50% confidence in 50.2% images while making the correct prediction in 81.2% of the cases with low confidence. Meaning the model is confident in the wrong predictions while making the correct predictions with low confidence; it is safe to assume that this model is failing for the "no-findings" class.

Comparing the results from the binary classifier in Figure 4.29b with the ones from Figure 4.29a shows that the "no-findings" class is responsible for the predictions with high confidence. The pathological images are being predicted with low-confidence.

Grad-CAM is a useful technique in deep learning that helps visually explain the results. This technique uses the gradients in the last convolutional layer (it could be inserted in any layer) to help localise and highlight parts of the images that contributed the most for each class the model predicted [57]. Figure 4.5.6 shows X-ray images the model correctly predicted for each pathological image. In each sub-figure 4.30a, 4.30b, 4.30c, 4.30d, the left image is the real image plotted with the bounding boxes given by the ChestX-ray14 designers, on the right is the activation colour-maps for each ground truth label. The red parts in the grad-cam images are the ones the model considers contributing the most. This visualisation confirms that the model is localising the correct parts of the images.




**(a)** Atelectasis/ (67.3-35.8)%

**(b)** Cardiomegaly/ (92.6-38.8)%

**(c)** Effusion/ (45.4-31.1)%

**(d)** Nodule / (73.3-44.8)%

**Figure 4.30:** Examples of images for each disorder localisation with the activation colour maps produced by Grad-CAM. The red regions are the parts in the image that contributed the most to the prediction. For each image legend, the labels represent the true class and the numbers the confidence with which the five multi-class (left) and joined binary (right) classifiers, predict the pathology.

Despite the model producing state-of-the-art results for cardiomegaly, effusion, and nodule, it cannot correctly identify non-pathological X-ray images, confusing supposedly healthy images with effusion.

### 4.5.7 Testing new models

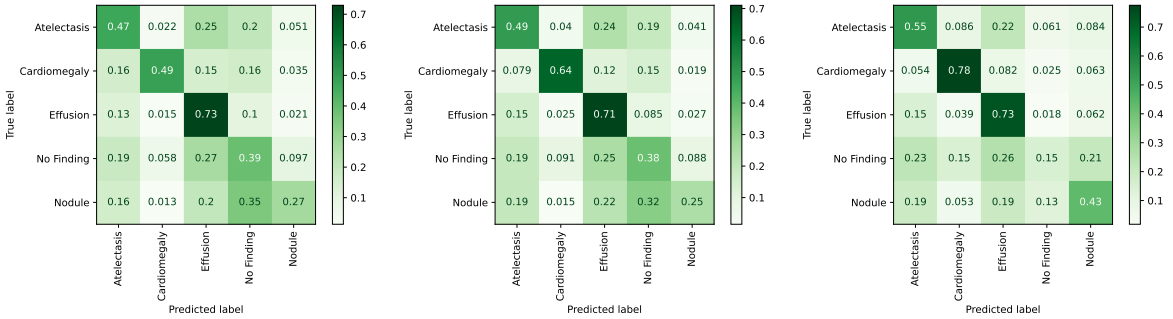
Given the poor results, one of two things can happen: the chosen model does not fit the data, or there is a problem with the dataset. In order to evaluate the first proposition, three new models were run with new architectures and weights in the loss function. The ResNet18 architecture was run with no class weights, the ResNet34 was run using a new formula for the class weights given by $class\_weights = \frac{1-\beta}{1-\beta_{[}n1,n2,...,n_c]}$ [58], and the ResNet101 was with the same class weights as in Section 4.1. If there is some improvement, the previous model was poorly chosen; if it behaves the same way, it may suggest some problems with the dataset.

Firstly, when training this new model, an improvement in the metrics was noticed. In all three new models, the accuracy and F1score increased by almost 7% and 2%, respectively. Table 4.9 shows this improvements. All of them revealed better results, especially the ResNet34 model. For this comparison to be meaningful, one has to look at the confusion matrices in Figure 4.31. For the model that uses the ResNet18, only the nodule and "no-findings" showed some improvement while the other classes got worse, which shows the improvement of metrics was mostly due to the "no-findings" class. In

**Table 4.9:** F1score and accuracy for all the best valid loss of the new models run. Using ResNet34 with the new class weights for the loss function seems to improve.

| Model | Epoch | Valid Loss | Accuracy(%) | F1score(%) |
|---|---|---|---|---|
| ResNet18 no class weights | 136 | 1.120 | 62.0 | 55.9 |
| ResNet34 new class weights | 165 | 1.145 | 62.6 | 57.5 |
| ResNet101 same class weights | 31 | 1.161 | 57.2 | 56.0 |

the supposed new best model, the ResNet34 with new weights, the no-finding class showed some improvements, but all the other classes, except for the atelectasis class, got worse. It only leaves the ResNet 101; this confusion matrix almost looks identical to the one in Figure 4.28a, meaning it shows a great value of confusion in the same "no-findings" class, this suggests the problem can be in the dataset, and the model has a reason to be confused, meaning there could be some incorrect ground truths, confusing images, images that should not be present or images that have no detail. Unfortunately, these problems are common when using medical databases.




**(a)** ResNet18 **(b)** ResNet34 **(c)** ResNet101

**Figure 4.31:** Confusion matrices for all the models in Table 4.9.

#### *4.5.7.1 Dataset problems*

The labels in the ChestX-ray14 [3] were extracted using natural language processing techniques from radiologists' reports. The dataset labels have an accuracy of 90%, so some images are mislabelled. This dissertation used a loss function with label smoothing to mitigate the errors in the labels, but more was needed to minimise all the errors in the dataset. Furthermore, the "no-findings" does not assure the non-presence of pathology in the images; the image's poor quality might prevent the radiologist and machine learning models from accurately recognising a pathology.

The previous results suggest a higher confusion in the "no-findings" images with effusion, which necessitates additional dataset analysis.

Images where the model failed to predict "no-findings" were picked up for further examination. This step helps to detect possible poor-quality images in the dataset. Figure 4.32 shows two examples where the "no-findings" images were mislabelled as effusion with the probability in the five multi-class models. Figure 4.32a shows an anomaly in the dataset, which displays a lateral chest X-ray image (the dataset should only contain frontal-view X-rays images). It is acceptable that the model produced the incorrect prediction since it was not designed to accomplish this. Figure 4.32b shows an image where the focus is not on the lungs. The image's poor quality reflects the probability that the model made the prediction; it is only 32% confident in this prediction.

The images from this dataset were extracted from a hospital where followed-up X-ray scans are standard. One thing that was also noticed was the frequency of the same patients in the testing

**(a)** "No-findings"/effusion /45.6% **(b)** "No-findings"/effusion/32.2%

**Figure 4.32:** Two examples of faulty images in the dataset. In (a) there is a lateral X-ray view; the dataset should only contain frontal views. In (b) there is an image where the focus is not on the lungs. The legend must be read as true label/ predicted label/ probabilities for the predicted label

dataset. The patient with ID 10007 has 72 images in this dataset. The five multi-class classifiers for this patient predicted 0 "no-finding" images, yet 45 images belong to this class. Additionally, the algorithm predicted 53 effusion pictures when there were only 21. There are two examples of this patient in Figure 4.33. In the left (Figure 4.33a), the model correctly predicts as effusion. On the right (Figure 4.33b) is the followed-up image where the truth label is "no-findings", but the model predicts it as effusion. When comparing the two images, Figure 4.33b where the model failed to predict seams to demonstrate more signs of effusion than the right image (Figure 4.33a). The image appears with white opacity obscuring the lower lungs; this could be a case of a wrong assigned label or a poorly taken X-ray image.

**(a)** Effusion/ Effusion / 63.2% **(b)** "No-findings"/ Effusion / 58.2%

**Figure 4.33:** Images for the patient with ID 10007. (a) shows an image where the model correctly labels it as effusion and (b) where the model failed to label it as "no-findings". The legend must be read as: true label/ predicted label/ probabilities for the predicted label.

The insertion of X-rays that are not of the chest region, lateral or oblique views of the chest area, and the chest not being present are examples of low-quality X-ray images that are frequently seen in medical databases. Besides, this database contains images taken with portable X-rays of patients in supine views, which tend to be challenging even for radiologists [59].

One of the main reasons why the development of pathology detectors using machine learning techniques has been delayed is the unavailability of high-quality and freely accessible medical image datasets [1]. Good results in the literature are achieved for either small datasets or ones where the dataset was cleaned [2]. However, this last step requires the radiologist and the machine learning community to better inspect at the images and decide on the good ones. In large datasets, this consumes a significant amount of time, and applying these models to real cases is not feasible because X-ray images will always have poor quality in some instances. If the models cannot detect these cases, it could lead to a wrong diagnosis affecting the patient's health. Nonetheless, it could be helpful for students to learn to identify some pathologies or for second-hand opinions.

CHAPTER 5

# Conclusion

The machine learning community has long been eager to develop algorithms to assist health professionals. Due to a lack of radiologists, there has been a high demand for pathological X-ray image detectors in developing countries. However, because machine learning algorithms demand massive, reliable, widely accessible databases, the passage from papers to production has been challenging. This issue was mitigated with the release of the ChestX-ray-14 database, which contains 112 120 frontal view X-ray pictures.

Two classifiers were developed in this dissertation using ResNet34 pre-trained on ImageNet and then trained on X-ray images to regognise and classify the following pathologies: atelectasis, cardiomegaly, effusion, nodule, and images labelled as "no-findings". The first one consisted of a multi-class classic classifier that which achieved an AUC of 0.73 for atelectasis, 0.91 for cardiomegaly, 0.82 for effusion, 0.67 for "no-findings" and 0.74 for nodule. The second approach was built using binary classifiers that classify images as pathological vs no pathological for each disease. The four binary classifiers were joined and asked to make predictions on all the images in the dataset. The predictions made by the four classifiers were then normalised, resulting in only five probabilities for each class, the highest of which represents the final prediction. The achieved AUC scores for this classifier were: 0.73 for atelectasis, 0.90 for cardiomegaly, 0.78 for effusion, 0.67 for "no-findings" and 0.74 for nodule. The results for both classifiers are not too far off compared with the literature, sometimes achieving better results. However, the best results in the literature were for models that disregarded poor-quality images and for different testing datasets, which makes it hard for a direct comparison.

Both classifiers showed poor results in classifying non-pathological images, correctly classifying less than 37% of the images and misclassifying many of these images as effusion. The analysis showed that in the no-finding images, the classifier has high confidence in the misclassifications and, in contrast, low confidence in the correct one. Due to this, it was necessary to gather the images for this class and thoroughly examine them to determine if the model had any justification for generating incorrect predictions. It was verified the existence of images with poor quality due to contrast, brightness and sharpness levels and lack of focus on the lungs, besides the existence of X-rays of the lungs that should not be present in the dataset.

This dissertation shows a promising path for the use of ResNet architectures in the detection of common chest pathologies in X-ray images. It achieved state-of-the-art-like results in cardiomegaly and effusion classification. Unfortunately, having large, high-quality datasets is the most crucial factor in getting good results which did not occur in this situation.

# Bibliography


[1] J. P. Cohen, P. Bertin, and V. Frappier, "Chester: A web delivered locally computed chest x-ray disease prediction system," *CoRR*, vol. abs/1901.11210, 2019. arXiv: 1901.11210. [Online]. Available: http://arxiv.org/abs/1901.11210.

[2] C. Bose and A. Basu, *Classification of covid-19 from cxr images in a 15-class scenario: An attempt to avoid bias in the system*, 2021. DOI: 10.48550/ARXIV.2109.12453. [Online]. Available: https://arxiv.org/abs/2109.12453.

[3] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2017, pp. 3462–3471.

[4] P. Rajpurkar *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *CoRR*, vol. abs/1711.05225, 2017. arXiv: 1711.05225. [Online]. Available: http://arxiv.org/abs/1711.05225.

[5] WebMD, *Respiratory system: Parts, function, and diseases*, https://www.webmd.com/lung/how-we-breathe, accessed October 2022.

[6] S. M. Art, *Smart - servier medical art*, https://smart.servier.com/, accessed October 2022.

[7] D. Peroni and A. Boner, "Atelectasis: Mechanisms, diagnosis and management," *Paediatric Respiratory Reviews*, vol. 1, pp. 274–278, 3 Sep. 2000, ISSN: 1526-0542. DOI: 10.1053/PRRV.2000.0059.

[8] WebMD, *Atelectasis: Types, causes, symptoms, treatment*, https://www.webmd.com/lung/atelectasis-facts, accessed October 2022.

[9] M. Clinic, *Enlarged heart - symptoms and causes - mayo clinic*, https://www.mayoclinic.org/diseases-conditions/enlarged-heart/symptoms-causes/syc-20355436, accessed October 2022.

[10] WebMD, *What is an enlarged heart (cardiomegaly)?* https://www.webmd.com/heart-disease/guide/enlarged-heart-causes-symptoms-types, accessed October 2022.

[11] C. Clinic, *Pleural effusion: Symptoms, causes, treatments*, https://my.clevelandclinic.org/health/diseases/17373-pleural-effusion-causes-signs--treatment, accessed October 2022.

[12] WebMD, *Pleural effusion - causes, symptoms, types, and treatments*, https://www.webmd.com/lung/pleural-effusion-symptoms-causes-treatments, accessed October 2022.

[13] C. Clinic, *Lung nodules (pulmonary nodules): Diagnosis, causes & treatment*, https://my.clevelandclinic.org/health/diseases/14799-pulmonary-nodules, accessed October 2022.

[14] S. S. Alghamdi, I. Abdelaziz, M. Albadri, S. Alyanbaawi, R. Aljondi, and A. Tajaldeen, "Study of cardiomegaly using chest x-ray," *Journal of Radiation Research and Applied Sciences*, vol. 13, pp. 460–467, 1 Jan. 2020, ISSN: 16878507. DOI: 10.1080/16878507.2020.1756187.

[15] T. Aach, U. W. Schiebel, and G. Spekowius, "Digital image acquisition and processing in medical x-ray imaging," *Journal of Electronic Imaging*, vol. 8, no. 1, pp. 7–22, 1999. DOI: 10.1117/1.482680. [Online]. Available: https://doi.org/10.1117/1.482680.

[16] "Recent development in x-ray imaging technology: Future and challenges," *Research*, vol. 2021, pp. 1–18, Dec. 2021, ISSN: 26395274. DOI: 10.34133/2021/9892152. [Online]. Available: /pmc/articles/PMC8724686/%20/pmc/articles/PMC8724686/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8724686/.

[17] H. stuff works, *How x-rays work | howstuffworks*, https://science.howstuffworks.com/x-ray.htm, accessed October 2022.

[18] *Chest x-ray (radiography)*, https://www.radiologyinfo.org/en/info/chestrad, accessed October 2022.

[19] A. Murphy, "Chest (pa view)," *Radiopaedia.org*, May 2016. DOI: 10.53347/RID-44853.

[20] K. H. Carlsen, S. Crowley, and B. Smevik, "Atelectasis," *Kendig's Disorders of the Respiratory Tract in Children*, 1027–1033.e1, Jan. 2019. DOI: 10.1016/B978-0-323-44887-1.00070-5.

[21] B. Botz and J. Jones, "Pleural effusion," *Radiopaedia.org*, May 2009. DOI: 10.53347/RID-6159. [Online]. Available: http://radiopaedia.org/articles/6159.

[22] Y. Weerakkody and J. Jones, "Pulmonary nodule," *Radiopaedia.org*, Jul. 2010. DOI: 10.53347/RID-10187. [Online]. Available: http://radiopaedia.org/articles/10187.

[23] Y.-B. Tang, Y.-X. Tang, J. Xiao, and R. M. Summers, "Xlsor: A robust and accurate lung segmentor on chest x-rays using criss-cross attention and customized radiorealistic abnormalities generation," in *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*, M. J. Cardoso *et al.*, Eds., ser. Proceedings of Machine Learning Research, vol. 102, PMLR, Aug. 2019, pp. 457–467. [Online]. Available: https://proceedings.mlr.press/v102/tang19a.html.

[24] L. Seyyed-Kalantari, G. Liu, M. McDermott, I. Y. Chen, and M. Ghassemi, "Chexclusion: Fairness gaps in deep chest x-ray classifiers," in *Biocomputing 2021*. 2021, pp. 232–243. DOI: 10.1142/9789811232701_0022. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789811232701_0022.

[25] A. Elkins, F. F. Freitas, and V. Sanz, *Developing an app to interpret chest x-rays to support the diagnosis of respiratory pathology with artificial intelligence*. DOI: 10.48550/ARXIV.1906.11282. [Online]. Available: https://arxiv.org/abs/1906.11282.

[26] A. M. DSouza, A. Z. Abidin, and A. Wismüller, "Automated identification of thoracic pathology from chest radiographs with enhanced training pipeline," in *Medical Imaging 2019: Computer-Aided Diagnosis*, H. K. Hahn and K. Mori, Eds., SPIE, Mar. 2019. DOI: 10.1117/12.2512600. [Online]. Available: https://doi.org/10.1117%2F12.2512600.

[27] *What is artificial intelligence: Definition and sub-fields of ai*, https://www.softwaretestinghelp.com/what-is-artificial-intelligence/, accessed January 2022.

[28] M. Haenlein, A. K. .-. C. management review, and undefined 2019, "A brief history of artificial intelligence: On the past, present, and future of artificial intelligence," *journals.sagepub.com*, vol. 61, pp. 5–14, 4 Aug. 2019. DOI: 10.1177/0008125619864925. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/0008125619864925.

[29] *Artificial intelligence (ai) – what it is and why it matters | sas*, https://www.sas.com/en_us/insights/analytics/what-is-artificial-intelligence.html, accessed January 2022.

[30] E. Commission *et al.*, *AI watch : defining artificial intelligence : towards an operational definition and taxonomy of artificial intelligence*. Publications Office, 2020. DOI: doi/10.2760/382730.

[31] *Artificial intelligence, machine learning, deep learning and more | sas*, https://www.sas.com/en_us/insights/articles/big-data/artificial-intelligence-machine-learning-deep-learning-and-beyond.html#/, accessed January 2022.

[32] *The history of artificial intelligence - science in the news*, https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/, accessed January 2022, Jan. 2022.

[33] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, First, N. Tache, Ed. O'Reilly Media, 2017, ISBN: 9781491962299.

[34] R. Bonetto and V. Latzko, "Chapter 8 - machine learning," in *Computing in Communication Networks*, F. H. Fitzek, F. Granelli, and P. Seeling, Eds., Academic Press, 2020, pp. 135–167, ISBN: 978-0-12-820488-7. DOI: 10.1016/B978-0-12-820488-7.00021-9.

[35] A. Burkov, *The Hundred-Page Machine Learning Book,* Andriy Burkov, 2020, ISBN: 1999579518.

[36] J. Howard and S. Gugger, *Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD*. O'Reilly Media, Incorporated, 2020, ISBN: 9781492045526. [Online]. Available: https://books.google.no/books?id=xd6LxgEACAAJ.

[37] *Focal loss: A better alternative for cross-entropy | by roshan nayak | towards data science*, https://towardsdatascience.com/focal-loss-a-better-alternative-for-cross-entropy-1d073d92d075, accessed August 2022.

[38] *What is label smoothing?. a technique to make your model less... | by wanshun wong | towards data science*, https://towardsdatascience.com/what-is-label-smoothing-108debd7ef06, accessed August 2022.

[39] I. C. Education, *What is gradient descent? | ibm*, https://www.ibm.com/cloud/learn/gradient-descent, accessed September 2022.

[40] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2014. DOI: 10.48550/arxiv.1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980v9.

[41] *3.3. metrics and scoring: Quantifying the quality of predictions — scikit-learn 1.1.2 documentation*, https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score, accessed August 2022.

[42] *Sklearn.metrics.average*$_p$*recision*$_s$*core*$^-$*scikit* − *learn*1.1.2*documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html#sklearn.metrics.average_precision_score, accessed August 2022.

[43] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science Business Media, 2022, ISBN: 978-3-030-34372-9.

[44] K. Zhou, H. Greenspan, and D. Shen, *Deep learning for medical image analysis*. Academic Press, 2017.

[45] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[46] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[47] J. Howard and R. Thomas, *Fastai- github*, https://github.com/fastai, accessed August 2022.

[48] J. Howard and S. Gugger, "Fastai: A layered api for deep learning," *Information (Switzerland)*, vol. 11, 2 Feb. 2020. DOI: 10.3390/info11020108. [Online]. Available: http://arxiv.org/abs/2002.04688%20http://dx.doi.org/10.3390/info11020108.

[49] S. Gugger, *Another data science student's blog – the 1cycle policy*, https://sgugger.github.io/the-1cycle-policy.html, accessed October 2022, 2018.

[50] Y. Weerakkody, *Pulmonary infiltrates | radiology reference article | radiopaedia.org*, https://radiopaedia.org/articles/pulmonary-infiltrates-1, accessed September 2022.

[51] H. S. Patterson and D. N. Sponaugle, "Is infiltrate a useful term in the interpretation of chest radiographs? physician survey results," *Radiology*, vol. 235, pp. 5–8, 1 Apr. 2005, ISSN: 0033-8419. DOI: 10.1148/RADIOL.2351020759. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/15798161/.

[52] $Sklearn.model_selection.stratifiedshuffflesplit^{-}scikit - learn1.1.2documentation$, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html, accessed September 2022.

[53] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017. arXiv: 1711.05101. [Online]. Available: http://arxiv.org/abs/1711.05101.

[54] L. N. Smith, *Cyclical learning rates for training neural networks*, 2015. DOI: 10.48550/ARXIV.1506.01186. [Online]. Available: https://arxiv.org/abs/1506.01186.

[55] Y. Xie and D. Richmond, "Pre-training on grayscale imagenet improves medical image classification," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Sep. 2018.

[56] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, *Transfusion: Understanding transfer learning for medical imaging*, 2019. DOI: 10.48550/ARXIV.1902.07208. [Online]. Available: https://arxiv.org/abs/1902.07208.

[57] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Oct. 2019. DOI: 10.1007/s11263-019-01228-7. [Online]. Available: https://doi.org/10.1007%2Fs11263-019-01228-7.

[58] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2019.

[59] A. Murphy, "Chest (supine view)," *Radiopaedia.org*, May 2016. DOI: 10.53347/RID-45309. [Online]. Available: http://radiopaedia.org/articles/45309.