Politecnico di Milano

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science – Computer Science and Engineering

A Machine Learning approach for automatic disease detection in Chest X-Rays

Supervisor
Pier Luca Lanzi

Co-Supervisors
Daniele Loiacono
Edoardo Giacomello

Candidate
Luca Nassano – 893812

Academic Year 2018 – 2019

Ringraziamenti

Un immenso grazie alla mia famiglia. A mamma e papà, per avermi dato l'opportunità di arrivare fino a qui, per avermi spronato ogni giorno a dare il meglio di me e per avermi insegnato tutto, grazie per aver fatto di me la persona che sono oggi. Ai miei fratelli, Andrea e Filippo, per aver condiviso con me me tutti questi anni, fatti di alti e bassi, grazie per esserci sempre stati. Il vostro sostegno è stato fondamentale. Vi voglio bene.

A tutte persone che son state vicine a me e alla mia famiglia in questi anni: Gianni, Enrico, Silvio, Francesca e Luisa, non dimenticherò mai il vostro supporto.

Ai miei amici. A Emanuel per aver condiviso con me questo viaggio, un traguardo dopo l'altro. A Riccardo, Marco ed Erik, per aver reso indimenticabili questi anni.

Al professor Loiacono, al professor Lanzi e al Dr. Giacomello, per la loro disponibilità e per avermi aiutato, in questi mesi, a portare a termine questo lavoro.

A tutte le persone che non ho menzionato ma che mi sono state accanto durante questo percorso, grazie ad ognuno di voi per aver contribuito ad arrivare fino a qui.

Abstract

Chest X-Rays, due to their simplicity and non-invasive nature, are one of the most common tools used for the detection of many diseases. During the last years the scientific community published different datasets containing CXR images, along with side information related to the presence or absence of different pathologies. This work studies the problem of analyzing them, with the aim of building an automatic system able to check whether a patient suffer from a given disease or not. Moreover, it tries to overcome the so called black-box problem, an issue connected to many machine learning applications, related to the fact that they are able to provide a decision but they're not capable of giving an explanation behind such decision. This work tries to solve it by producing, along with the prediction, a heatmap highlighting the region that most probably is affected by the disease and a bounding box surrounding it. In particular, in this work we investigate different approaches, ranging from Convolutional Neural Network, widely employed in other related works, to the less used Random Forest, trained using a low dimension representation of the input, the so called embeddings. We also propose a novel technique to combine different predictions, that exploit the uncertainty of the single model to assign it a proper weight during the aggregation phase. Although we weren't able to surpass the performance of some related works, we obtained a mean AUROC of 0.902 over five different pathologies, showing that good results can be achieved also by models that are computationally more efficient and require much less time to be trained with respect to CNNs.

Estratto in lingua Italiana

Le radiografie del torace, data la loro semplicità e natura non invasiva, sono uno degli strumenti più comunemente utilizzati per la rilevazione di molte malattie. Negli ultimi anni la comunità scientifica ha pubblicato diversi dataset contenenti radiografie toraciche, accompagnate da informazioni aggiuntive riguardo la presenza o l'assenza di diverse patologie. Questo lavoro si pone il problema di analizzarle, con l'obiettivo di costruire un sistema in grado di verificare automaticamente se un paziente soffre o meno di una determinata patologia. Inoltre, cerca di risolvere il cosiddetto problema della scatola nera, relativo a molte applicazioni di Machine Learning, legato al fatto che spesso queste ultime sono in grado di prendere una decisione ma non sono in grado di fornire una spiegazione riguardante i motivi che hanno guidato quella scelta. Questo lavoro cerca di risolvere questo problema, producendo, in aggiunta alla previsione, una mappa di calore che evidenzi la regione affetta dalla malattia, insieme ad un riquadro che la circondi. In particolare, in questo studio indaghiamo diversi approcci, partendo dalle reti neurali di convoluzione, ampiamente impiegate in altre opere correlate a questa, sino alle meno utilizzate Random Forests, addestrate utilizzando i cosiddetti embedding, una rappresentazione dell'input di dimensioni ridotte. Proponiamo inoltre una nuova tecnica per combinare le diverse previsioni, che sfrutta l'incertezza del singolo modello per assegnargli un peso adeguato durante la fase di aggregazione. Sebbene non siamo stati in grado di superare le prestazioni di alcuni lavori simili a questo, abbiamo ottenuto un AUROC medio di 0.902 calcolato su cinque diverse patologie, dimostrando che si possono ottenere buoni risultati anche con modelli che

sono computazionalmente più efficienti e che richiedono molto meno tempo per essere addestrati rispetto alle reti neurali di convoluzione.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

CXR is one of the most common tool used for the detection of many diseases. It's an easy, fast and non invasive medical test that employs ionizing radiation in order to produce a chest image that can be used to diagnose many conditions involving heart, lungs, airways, blood vessels and the bones of the spine and chest.

Despite the simplicity of the test, reading chest X-ray images may be a challenging task requiring careful observation and knowledge of anatomical principles, physiology and pathology. Because of this a highly qualified figure, the radiologist, is often needed in place of a doctor. Even though this seems not be a problem for many countries, it could be a big issue for the poorest one, where the lack of medical resources and personnel could be fatal. In Africa's 47 countries, for example, there is a deficit amounting to 2.4 million doctors and nurses [22]. For these populations a fast and accurate diagnosis is fundamental in order to guarantee timely access to treatments. Moreover, the average time it takes a well trained radiologist to read a radiography is about 1-2 minutes. However, considering that an hospital can generate hundreds or even thousands of CXRs every day, the overall amount of time is massive.

It would be very useful, then, to have an automated system able to support the work of medical staff. The aim is not to replace human's effort, but to support it, improving the quality of the diagnosis and speeding up them. Being able to provide a fast and accurate evaluation of the CXR could be extremely useful in many cases in order to provide a prompt therapy to the patient.

The increment in the volume of available data of the last years and the recent improvements in the field of Machine Learning (ML) and, in particular, of Deep Learning (DL) made possible to develop a lot of automated systems able to support and enhance the healthcare. Specifically, the realization of an autonomous system

able to read a CXR and provide a reliable diagnosis had become a feasible objective.

1.1 Scope

In this work we tackle the problem of analyzing CXRs using not only CNNs, one of the most used tool to examine medical images, but also ML approaches,such as RF. We also investigate how these different methods can be combined in order to enhance the performance. The goal of this work, however, is not only to classify whether a given CXR is healthy or not, but also to give an explanation on why that decision has been taken. Indeed a big problem related to the field of Artificial Intelligence (AI) is that many algorithms work as a black box: they are fed with data and a result is produced, but we're often unsure about what happened in the middle and, in particular, we don't know which are the reasons that drove the model to take that decision. Although the process that generate an outcome could be less interesting for many scenarios, it is instead extremely important in medicine, where it's crucial to know why a patient is sick. In order to overcome this problem and make the prediction more explainable, we generate an heat map overlapping the radiography that highlights the area affected by the disease.

1.2 Thesis Structure

In Chapter 2 we will overview the related works that have been done in this field as well as the theoretical background needed to understand how we designed our system. Chapter 3 will formally present the problem and the content of the dataset that we have used and we'll show how the data have been preprocessed. We will also discuss about the related works that have been done involving the same dataset. Chapter 4 will describe the solution we developed to solve the problem. In Chapter 5 we will present the obtained results and they will be compared with the current state of the art. Finally, in the last chapter, we will give a general consideration about our work and we'll propose further approaches that can be implemented to improve it.

Chapter 2

Related Works And Theoretical Background

This chapter presents the related work that have been done in healthcare and introduces the theoretical background upon which this work is built, with the aim of providing the reader the knowledge he needs to better understand it.

2.1 Related Works

2.1.1 Artificial Intelligence in healthcare

The term AI was coined in 1956 during a workshop in Dartmouth College by John McCarthy, who defined it as «the science and engineering of making intelligent machines [...]» [19]. One of the first application of AI to healthcare dates back to 1970s, when a team of experts at Stanford University developed the system MYCIN [28]. The goal of the project was to create a tool able to identify bacteria causing infections and to recommend antibiotics based on patient's weight. MYCIN used a knowledge base composed by almost 600 rules and an inference engine that applied logical rules to it in order to deduce new information. Despite it was able to achieve results comparable to that of specialists, it was never used in practice, both because it required more computing power and memory than that available in most of the hospital at the time and also because it required a user to enter all relevant information about a patient by typing in responses to some questions that MYCIN posed.

During the 1980s and 1990s several other approaches have been explored in order to enhance the healthcare. Klaus-Peter Adlassnig [1] proposed, in 1980, a model to assist medical diagnosis based on L.A. Zadeh theory of Fuzzy Sets [38], well suited for medical applications because it can deal with the inherent ambiguity of medical data.

Basilio et al. [30] in 1998 combined Genetic Algorithms [10] and Bayesian Networks [23] to predict survival in patient with malignant skin melanoma.

But among all the different approaches that have been explored, the most used has been, judging by the volume of publications [32], ANNs, introduced for the first time by W.S. McCulloch and W. Pitts in 1943 [20]. They are a computing system inspired by animal's biological neural network, able to recognize patterns that are too complex or numerous for a human to extract. Because of their ability to recognise and exploit the intricate relationship among different variables, ANNs have been successfully used in many clinical scenarios.
One of the first work involving ANN applied to the medical field has been proposed by W.G Baxt in 1990, when he realized a system for the diagnosis of myocardial infarction by means of an ANN. Since then many other application involving Neural Networks has been proposed, for example to identify epilepsy [36], appendicitis [24] and cancer [17].
During the last fifty years AI has been widely used for healthcare-related applications. Nowadays it's usage is related, but not limited, to disease diagnosis, medical images analysis, prognosis, drug interactions and genomics.

2.1.2 Convolutional Neural Networks in computer vision

In 1989 LeCun et al. [15] introduced one of the first Convolutional Neural Network, a special type of ANN particularly effective for the task of analyzing visual images. In their work, specifically, they used it to recognize handwritten zip code numbers. The novelty with respect previous approaches was that the learning process was fully automatic, in the sense that the features used to make the classification were learned by the network instead of being manually designed by a human being. Since their introduction, CNNs have been widely used for many tasks. The most straightforward is that of image classification, in which an image is given as input to the network and it's assigned to a given class (Figure 2.1).

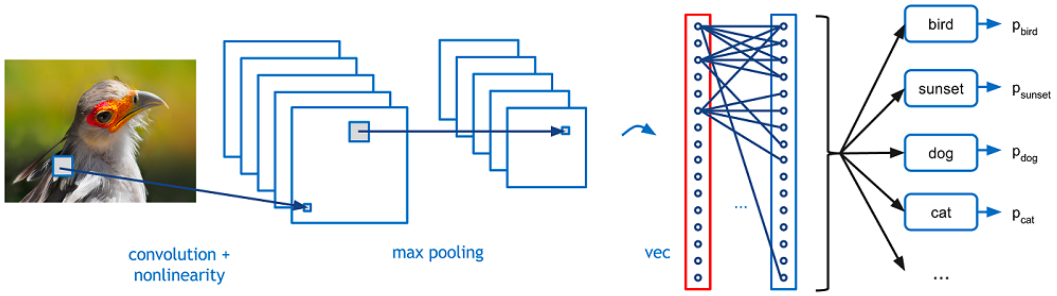


Figure 2.1: Image Classification using a CNN.

One of the most interesting peculiarity of CNNs applied to the task of image classification is their ability to explain which part of the picture they are looking at while assigning the class. This can be done by producing a Class Activation Map (CAM), a 2D grid of scores computed for every location of the input image, indicating how important is each location with respect to a given class. Figure 2.2 shows an example of CAMs produced while assigning the labels "cat" and "dog" to the image given as input. As we can see the network is looking at the correct part of the image, paying attention at the top area to classify the dog and at the bottom one to classify the cat.

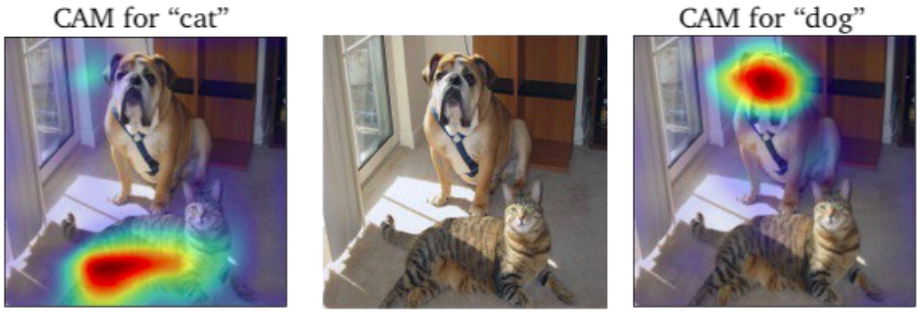


Figure 2.2: Example of Class Activation Map [27].

2.1.3 Deep Learning in Chest X-Ray Diagnosis

During the last years the problem of CXR diagnosis has been tackled multiple times using different approaches. One of the major drawbacks of DL is that it requires a massive amount of data in order to achieve good results. The first public dataset containing a significant amount of labeled radiography is "ChestX-ray8", that has been release in 2017 by the National Institute of Health [37]. It is composed by 108,948 frontal-view X-Ray images coming from 32,717 unique patients. Each radiography is associated with 8 different labels (subsequently increased to 14), each one referring

to a particular disease. Among the large amount of work that has been done by the scientific community using this dataset, it worth mentioning that of Rajpurkar et al. [26], able to achieve performance statistically significantly higher than radiologist. In 2019 two new dataset has been released: Chexpert [12], composed by 224,316 chest radiographs of 65,240 patients, and MIMIC-CXR-JPG [13], made of 377,110 images. CXRs from the two dataset are provided with the same 14 labels, that have been extracted from radiology report using Natural Language Processing (NLP).

2.2 Neural Networks

Fully Connected Neural Networks (FCNNs) are a computing system able to catch the relationship binding an input to an output. They are based on a Supervised Learning paradigm: in order to learn the function mapping an input to its corresponding output, the network must be trained using a lot of examples composed by both the input and the desired output.

A FCNN is made of a series of hidden layers, each one composed by a given number of elementary units, called artificial neurons, that are fully connected to the ones belonging to the previous layer. When the network receives an input, it pass through all the network and gets transformed by the artificial neurons, until it reaches the last layer, where an output is carried out (Figure 2.3).

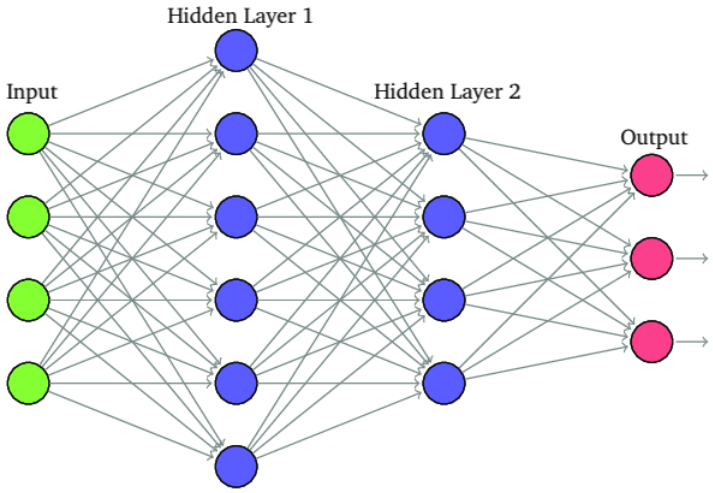


Figure 2.3: Fully Connected Neural Network

This model, however, is not suitable for image processing. Indeed, if we consider, for example, an image with shape $32 \times 32 \times 3$ pixels, a single neuron in the first hidden layer of a FCNN would have a number of connections equal to 32*32*3 = 3072. Although it seems to be a reasonable amount of weights, this approach does not scale well when we are dealing with larger images. As we will see in next chapters, our images will have a size of $224 \times 224 \times 3$, using FCNN would lead a single neuron to have 150,528 connections. Such a number of parameter is intractable and would lead to overfitting.

2.2.1 Convolutional Neural Network

Convolutional Neural Networks are a particular type of Neural Networks that have been extensively used over the past years, able to overcome the limits of FCNN discussed in section 2.2. CNNs have been successfully used for the first time by Yann Lecun for the task of recognizing handwritten digits [15]. The most important building block introduced with CNNs is the convolutional layer, designed to take advantage of the multi-dimensional structure of the input. Neurons in the first convolutional layer are not connected to every single pixel of the input image, but only to a small part of it, the so called receptive field. CNNs have a hierarchical structure: in the first layers the network extracts simple patterns, such as edges and curves that, in the following layers, are assembled to generate higher level features (Figure 2.4).

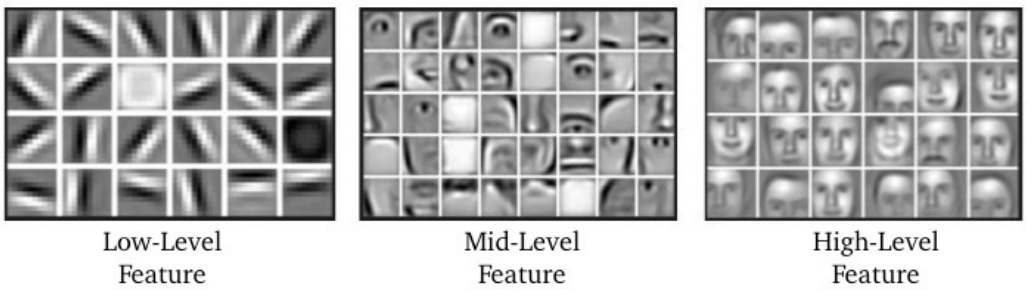


Figure 2.4: Feature hierarchy in a CNN

Feature extraction is accomplished by the so called kernels (also known as filter). Figure 2.5 shows an example of two kernels applied to an image: in the top left picture a vertical filter has been used, thus all the vertical lines in the original image gets enhanced, while the rest gets blurred. The top-right image, instead, shows what happen when an horizontal filter is applied. Of course kernels do not have to be defined manually, during the training phase the convolutional layers will automatically learn the most useful for the given task and the subsequent layers will learn to combine them into more complex pattern.

The mathematical operation that allows the network to extract the features from the input image is called convolution, that consist in sliding a kernel (that is basically a small matrix of number), across the whole dimensions of the input and compute the dot product between the entries of the filter and the input at any position. Convolution between image I and kernel w of size $k \times k$ can be computed as:

$$(I * w)[r, c] = \sum_{x=-k}^{k} \sum_{y=-k}^{k} w(x, y) I(r - x, c - y) \tag{2.1}$$

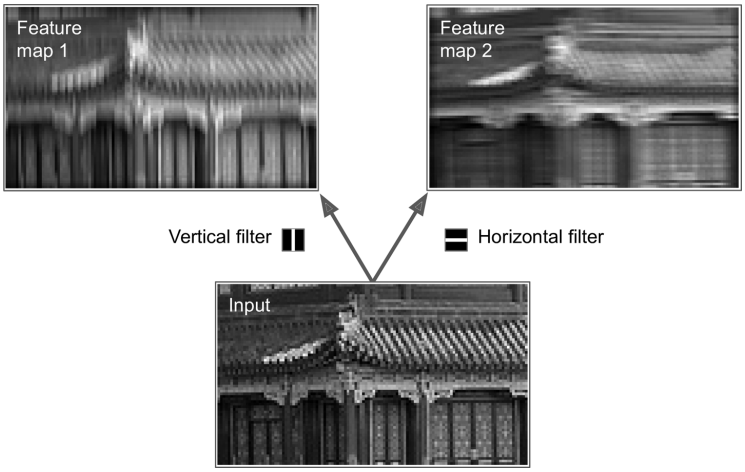


Figure 2.5: Example of filters applied to an image [8]

An example of convolution is shown in Figure 2.6, in which the value of the destination pixel has been computed as:

$$(4 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 1) + (0 \cdot 1) + (0 \cdot 0) + (0 \cdot 1) + (-4 \cdot 2) = -8 \tag{2.2}$$

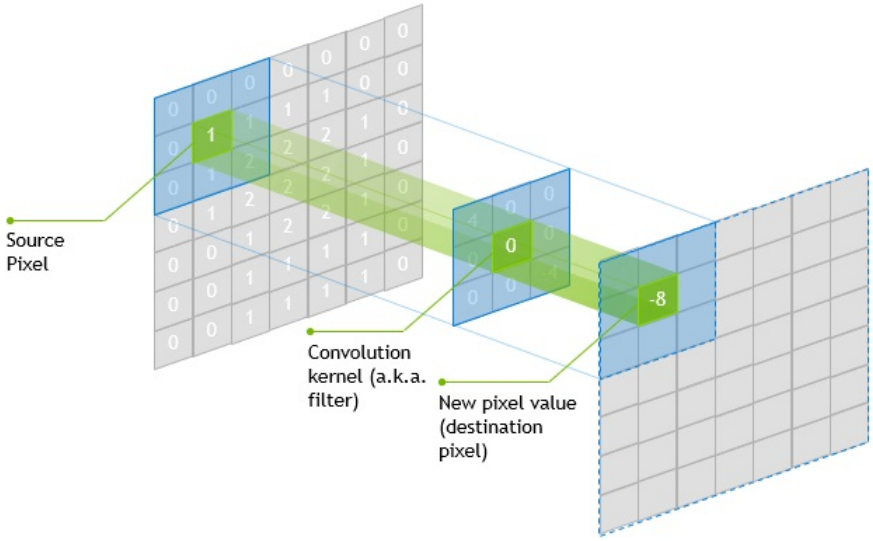


Figure 2.6: Example of convolution operation between an image and a kernel

Another widely used layer in CNNs is the Pooling Layer. Its function is to progressively decrease the spatial size of the representation in order to reduce the computational load, the memory usage and the number of parameter (preventing thereby the risk of overfitting). A pooling layer has no weights, what it does is to aggregate the input using an aggregation function, such as mean or max (Figure 2.7). A max pooling layer, moreover, introduces some level of invariance to small translations, a feature that could be useful for the task of image classification.

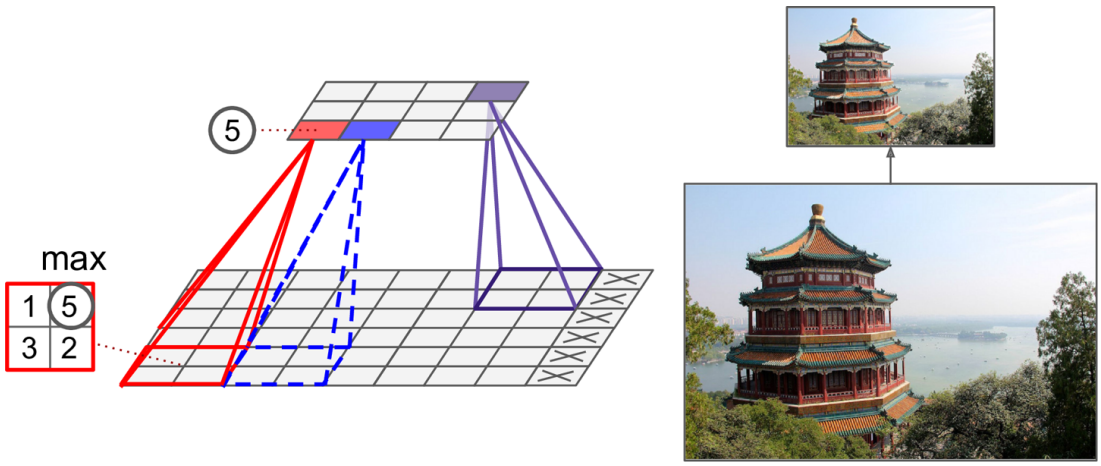


Figure 2.7: Example of Max Pooling

2.2.2 Class Activation Map

In 2016 Zhou et al. [39] shown that a CNN is able to identify exactly which region of an image is used for discrimination. The novelty with respect to previous works is that the network is only trained for solving the classification task, thus no bounding box annotation is provided as input to the network. The remarkable localization ability of CNN is lost when the fully connected layer is used for classification. Thanks to the work of Lin et al. [16], this problem can be overcome by means of a Global Average Pooling (GAP) layer, initially introduced to minimize the number of parameters, avoiding the usage of fully connected layer while maintaining high classification performance. Similarly to Pooling Layers, a GAP layer aims at reducing the spatial dimension of the input tensor. However GAP makes a more extreme type of dimensionality reduction, transforming a three dimensional input of size $h \times w \times d$ into a tensor with shape $1 \times 1 \times d$ (Figure 2.8).

In order to generate the CAM, we will make a weighted sum of the feature maps at the last convolutional layer. If we indicate with $f_k(x, y)$ the activation of unit k at spatial location (x, y) then the result of performing Global Average Pooling is given

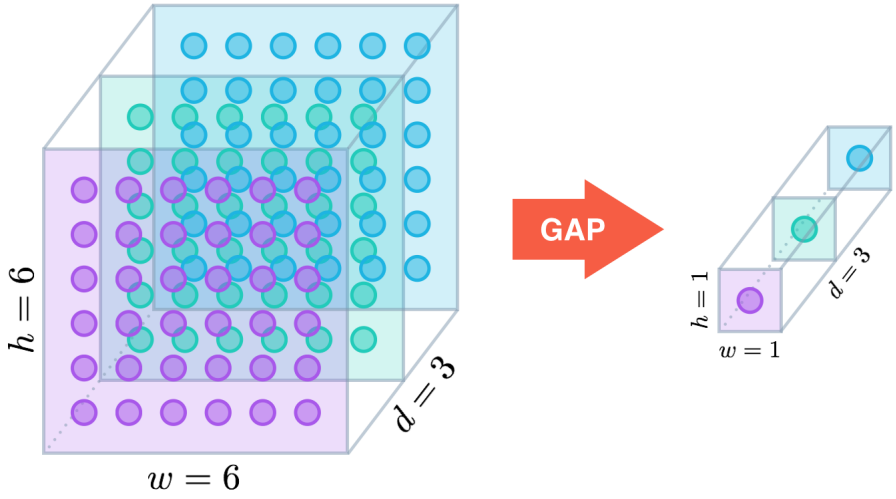


Figure 2.8: Example of Globale Average Pooling [29].

by:

$$F_k = \sum_{x,y} f_k(x, y) \tag{2.3}$$

Thus the input to the softmax for a given class will be:

$$S_c = \sum_k w_k^c F_k = \sum_{x,y} \sum_k w_k^c f_k(x, y) \tag{2.4}$$

Essentially w_k^c represent the importance of F_k for class c. Once S_c has been calculated, the probability for a class c can be computed using the softmax function:

$$P_c = \frac{exp(S_c)}{\sum_c exp(S_c)} \tag{2.5}$$

While the Class Activation Map for class c, instead, can be computed as :

$$M_c(x, y) = \sum_k w_k^c f_k(x, y) \tag{2.6}$$

So in order to extract the most important part of the image for a given class, we project back the weights of the output layer on the convolutional feature maps obtained from the last Convolution Layer. Image 2.9, taken from the work of Zhou et al. [27], shows how the CAM is computed by making a weighted sum of the last convolutional layer's features.

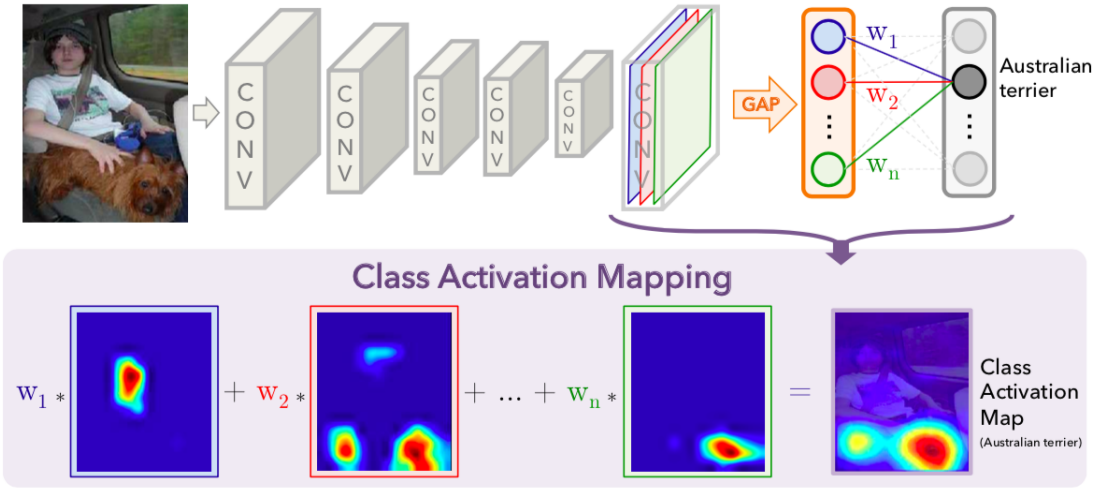


Figure 2.9: Class Activation Map generation.

2.2.3 Embedding

In the context of ANN a very useful techniques is Embedding. When dealing with images, we generally work with a massive amount of data. A simple colored image with resolution of 224×224 pixels is converted into a tensor having $224 \times 224 \times 3 = 150.528$ entries. Although CNNs are able to deal with such a large amount of data, the vast majority of algorithms can't process it in a reasonable amount of time. In order to overcome this issue we can rely on Embedding, a set of techniques used to reduce the input dimension. An embedding is a learned low dimensional vector representing the input. Once an embedding has been extracted, it somehow contains the same relevant information of the input but in a smaller and more manageable representation, suitable for being used by other kind of algorithms.

CNNs can be used to extract the low dimensional feature vector. In the hidden layers, in fact, the network learns a representation of the input that is subsequently used for classification. If we cut the network at one of the hidden layer, thus, we can extract a low dimension vector representing the the input (Figure 2.10).

2.3 Ensemble Learning

In social science there is a theory, called "The Wisdom of Crowd", stating that the aggregate decision made by a group of people will often be better than those of it's individual members [2]. A similar idea is exploited by Ensemble Learning, a technique that allows to improve machine learning results by combining several models. Figure 2.11 shows a typical workflow exploiting ensemble: several classifiers are trained and

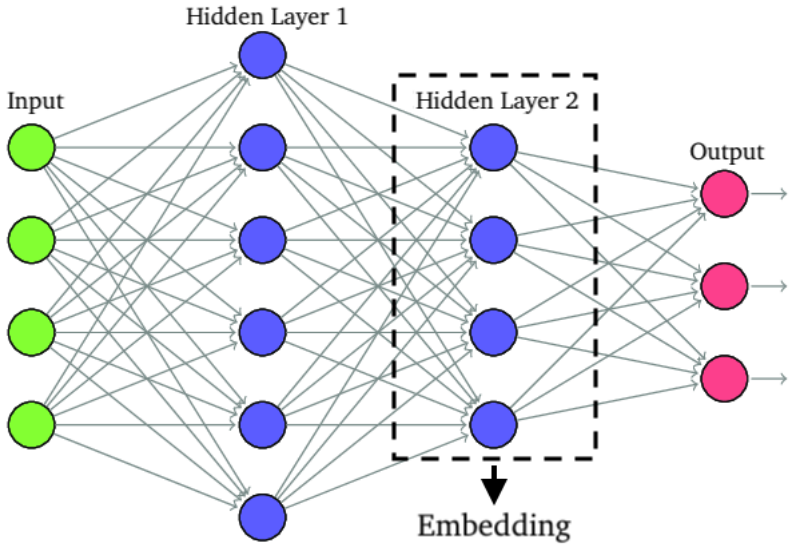


Figure 2.10: Embedding extraction from an ANN

then combined in order to carry out a single prediction that takes care of the opinion coming from all the base classifiers. The keypoint here is to find the best way to

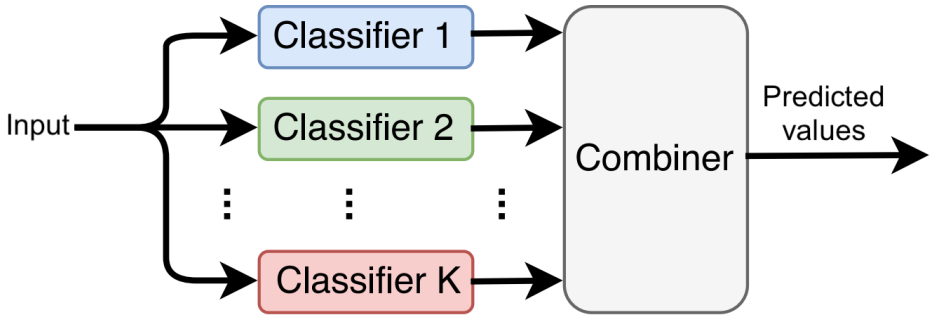


Figure 2.11: Ensemble Learning

combine all the predictions. There are different approaches, such as:

- **Ensemble Averaging**: the predictions that are carried out from the single experts are merged together using either a mere average or a weighted sum. Doing this helps reducing the variance of the output. For example, if we indicate as y_i the output of each base classifier, then the ensemble of the n experts can be computed as:

$$\bar{y}(\mathbf{x}, \alpha) = \sum_{i=1}^{n} \alpha_i y_i(\mathbf{x}) \tag{2.7}$$

- **Bootstrap Aggregating**: also known as Bagging [3], this technique aims at reducing the variance by averaging the output of multiple models built on different training set. More specifically, given a dataset D, each model if fitted

using a dataset D_i build by sampling uniformly and with replacement from D.

- **Boosting**: this aggregation approach aims at producing a strong predictor by incrementally building the weak classifiers. It's peculiarity is that each model instance is trained paying more attention to the samples that were misclassified by the previous predictor.

- **Stacking**: this approach combines multiple classification or regression models using a meta-classifier or a meta-regressor. Specifically, a training set is used to train the base classifiers, then the predictions that are carried from the classifiers are used as feature to train the meta-learner. The pseudocode for the general stacking algorithm is shown in figure 2.12

Algorithm Stacking

1: Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
2: Ouput: ensemble classifier H
3: *Step 1: learn base-level classifiers*
4: **for** $t = 1$ to T **do**
5: learn h_t based on D
6: **end for**
7: *Step 2: construct new data set of predictions*
8: **for** $i = 1$ to m **do**
9: $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_T(x_i)\}$
10: **end for**
11: *Step 3: learn a meta-classifier*
12: learn H based on D_h
13: return H

Figure 2.12: Stacking Pseudocode

2.4 Random Forest

Random Forests are an ensemble algorithm proposed by T.K. Ho in 1995 [35]. The idea behind RFs is to build a multitude of decision trees and combine their predictions. Decision Trees are a supervised-learning method used bot for classification and regression tasks. The algorithm builds a tree-like structure in which:

- An internal node is a test on an attribute.

- A branch represent an outcome of the test.

- A leaf represent a class label or a class label distribution.

Figure 2.13 shows an example of Decision Tree used to predict survivals on Titanic. A new instance is assigned to a class by following a path starting from the root and ending in a leaf node, testing a different feature at each internal node.

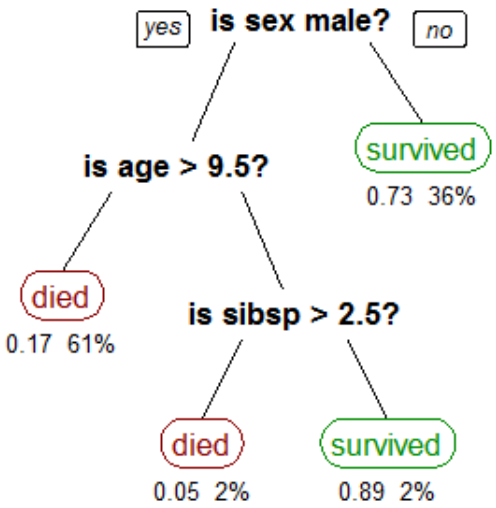


Figure 2.13: Decision Tree showing survival of passenger on Titanic [21]

Decision Trees have many advantages: they are very fast in making predictions and, differently from ANNs, they are a white box model, in the sense that they are easy to interpret and visualize. Indeed it's possible to easily check which are the features that most contribute to carry out a particular prediction. However, they also have some drawback. They often create over-complex structure that are not able to generalize well, thus leading to overfitting. Another big issue is that they are unstable, in the sense that a small variation in the training data may generate a totally different model. These issues can be overcome by resorting to Random Forest, that are basically an ensemble of Decision Trees based on Bagging, a technique seen in section 2.3. The main steps to build a Random Forest are the following:

- Build k training set D_i by sampling with replacement from D
- Learn tree T_i from D_i using only a random subset of the input variables.
- Save the tree as it is, without performing any pruning

Once the Random Forest has been created, the predictions computed by the trees are combined using:

- Majority Voting (in the case of Classification)

- Average (in the case of Regression)

When the number of trees is sufficiently large, Random Forest can prevent overfitting. Moreover they are an easy to use tool, having only two important parameters to tune (number of trees and percentage of variables for split).

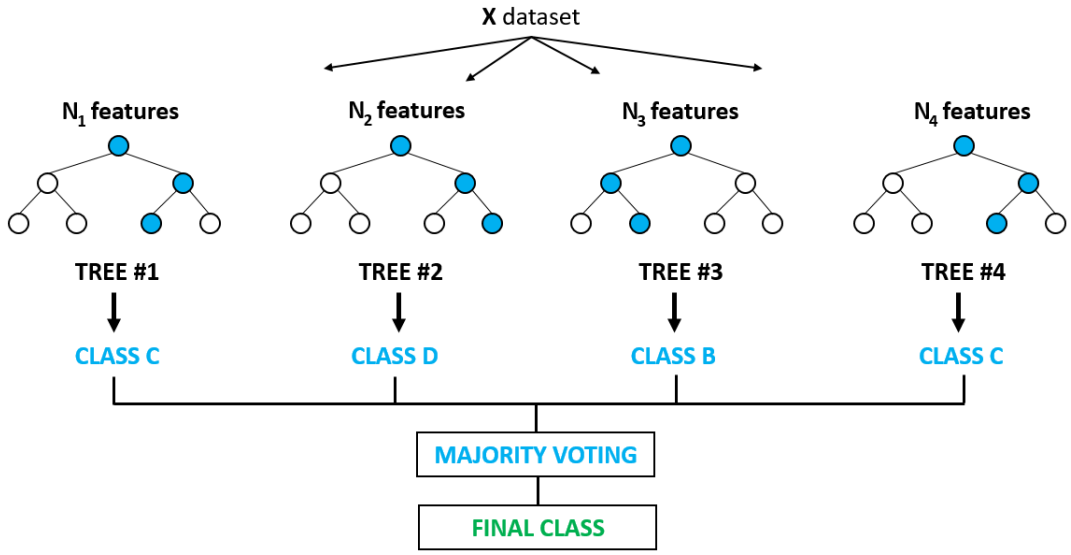


Figure 2.14: Example of Random Forest for classification task

2.5 Gradient Boosting

Gradient Boosting is a learning algorithm proposed in 1990 by Friedman [7]. As the name suggests, it's a technique based on boosting. The idea of this algorithm can be summarized by the following steps:

1. Learn a predictor.
2. Compute the error residual.
3. Learn to predict the residual
4. Goto point 2

So Gradient Boosting tries to repetitively leverage the patterns in the residuals to improve the successive predictions. The predictors used as weak leaerner in gradient boosting are decision trees. Note that, however, gradient boosting is a greedy algorithm and, thus, it can overfit the dataset quickly. A solution to this problem is that of constraining the trees, being the shallow trees, generally, less prone to overfitting.

2.6 Summary

The goal of this chapter was to give an general overview of the main techniques that will be used in the following chapters. We've seen what Artificial Neural Networks are and how they can be used in order to deal with images, both for classify them and localize interesting areas. We've also seen how multiple approaches can be combined together to enhance the global performance, with a particular attention on Random Forest. In the next chapter we will introduce the dataset and we'll describe the pre-processing phase.

Chapter 3

Chest X-Rays Automated Diagnosis

The goal of this chapter is to formally present the problem and introduce the dataset that we will use to train, evaluate and test the models that will be presented in the following chapter. In particular we want to give a description about the content of the dataset, its structure and also the pre-processing steps that have be done.

3.1 Problem Formulation

The focus of this work is to build a system able to make accurate diagnoses of 14 different diseases analyzing an input image containing a chest x-ray. Formally, we have a dataset $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}); i : 1, ..., N\}$ containing N CXRs, where each image $\mathbf{x}^{(i)}$ is associated with a set of label $\mathbf{y}^{(i)} \in \{0, 1\}^{14}$, where a value of 0 means that the corresponding disease is not present on the given input, while a value of 1, instead, means that it's present. We want our system to be able to predict, given a new instance $\mathbf{x}^{(i)}$, the corresponding output $\mathbf{y}^{(i)}$. To be more specific, our system will carry out a value corresponding to the probability, for each pathology, to be present or not on a given CXR. It is worth noting that this is not a multiclass classification problem, in which there is the assumption that a sample can be assigned to one and only one class, but it's a multilabel classification problem, in which a sample can be assigned to an arbitrary number of classes.

We also want our model to be able to produce a map highlighting the area of the image interested by the pathology and use it to automatically extract a bounding box around the disease region.

3.2 Data Source

In our work we've decided to use CheXpert (**Che**st e**Xpert**), a large dataset released in 2019 by Stanford University for CXR examination [12].

3.2.1 Content

The dataset is composed by 223,316 chest x-rays of 65,240 patients, collected by Stanford Hospital from October 2002 to July 2017. The dataset is offered at two quality levels: the high quality one has a size of 439GB and the images are stored as 16-bit PNG, while the small version has a size of 11GB and stores the chest x-rays using a 8-bit PNG format. In this work, due to computational reasons, we have decided to use the small version.
Each image is annotated with 14 labels, that have been extracted from text radiology report using an automatic rule-based labeler. The labelling process can be divided in three different phases:

- Mention Extraction: the *Impression* section of the radiology report, which summarizes the key finding in the radiology study, is analyzed and a list of mentions is extracted. The extracted mentions are those that match a large list of phrases that has been manually created by multiple expert radiologist.
- Mention Classification: this phase aims at assigning each mention to a label among:
  - Positive: the disease is confidently present
  - Negative: the disease is confidently absent
  - Uncertain: this value captures both the uncertainty in the radiologist diagnosis as well as the inherent ambiguity of the report.
- Mention Aggregation: the last phase aggregates the classification of all the mentions in order to generate a label for all the 14 observations. Each label is assigned a positive (1), negative (0) or uncertain (u) value if it has been explicitly mentioned in the report, while the (*blank*) value is assigned to the unmentioned observations.

The labeller has been tested by the Stanford Team against human labelled cases, showing that it is able to achieve very high perfomance.

The labels have a hierarchical structure, that has been provided along with the dataset and can be seen in Figure 3.1. As we can see, for example, there is a dependency between Lung Opacity and Edema, meaning that an X-Ray presenting Edema must also contain a Lung Opacity (but the contrary must not necessarily be true).

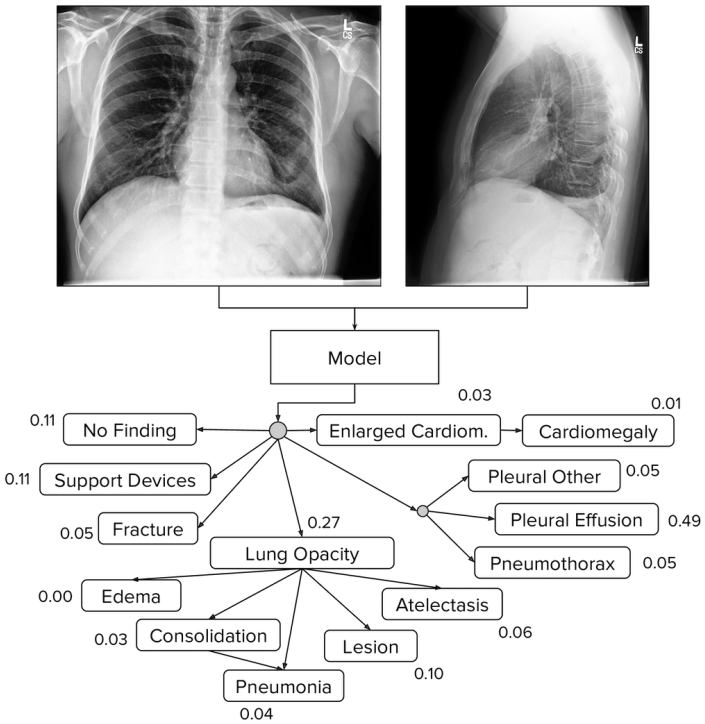


Figure 3.1: Pathologies hierarchy

This additional information can be exploited in order to enhance the quality of the predictions.
Table 3.1 gives a brief description of all the medical conditions that have been considered in the dataset.

Pathology Name	**Description**
No Finding	No pathology is assigned to a positive or uncertain label
Enlarged Cardiomediastinum	Abnormal enlargement of the cardiomediastinal silhouette
Cardiomegaly	Condition where the heart is enlarged.
Lung Lesion	Damaged portion of a lung.
Lung Opacity	Area that attenuates x-ray beams, making the lung appearing more opaque than the surrounding region.
Edema	Fluid accumulation in the tissue and air spaces of the lung.
Consolidation	Region of normally compressible lung tissue that has filled with liquid instead of air. The condition is marked by induration.
Pneumonia	Pneumonia is an infection that inflames the air sacs in one or both lungs.
Atelectasis	Collapse or closure of a lung resulting in reduced or absent gas exchange.
Pneumothorax	Condition in which air accumulates in the pleural sac, causing it to expand and compress the underlying lung.
Pleural Effusion	Excess of fluid that accumulates in the pleural cavity.
Pleural Other	Other pleural disease
Fracture	Partial or complete break in the continuity of a bone.
Support Device	Medical device aims at supporting or sustaining patient's life.

Table 3.1: Definition of all the diseases considered in the dataset

3.2.2 Structure

As mentioned in the previous section, the dataset contains 224,316 chest X-Rays extracted from 65,240 different patients. The details about the distribution of the different labels are reported in Table 3.2.

The Stanford Team also provided a set of manually labeled images that can be used to assess the performance of the predictions. We additionally cut out a subset of data that will be used to tune the models' hyperparameters. The dataset, moreover, contains two type of CXRs, the frontal-view and the lateral-view. Generally the latest one is performed as an adjunct to a frontal chest radiograph in cases where there is a diagnostic uncertainty. For this reason, the number of frontal images is much higher.

Pathology	**Positive (%)**	**Uncertain (%)**	**Negative (%)**
No Finding	16627 (8.86)	0 (0.0)	171014 (91.14)
Enlarged Cardiom.	9020 (4.81)	10148 (5.41)	168473 (89.78)
Cardiomegaly	23002 (12.26)	6597 (3.52)	158042 (84.23)
Lung Lesion	6856 (3.65)	1071 (0.57)	179714 (95.78)
Lung Opacity	92669 (49.39)	4341 (2.31)	90631 (48.3)
Edema	48905 (26.06)	11571 (6.17)	127165 (67.77)
Consolidation	12730 (6.78)	23976 (12.78)	150935 (80.44)
Pneumonia	4576 (2.44)	15658 (8.34)	167407 (89.22)
Atelectasis	29333 (15.63)	29377 (15.66)	128931 (68.71)
Pneumothorax	17313 (9.23)	2663 (1.42)	167665 (89.35)
Pleural Effusion	75696 (40.34)	9419 (5.02)	102526 (54.64)
Pleural Other	2441 (1.3)	1771 (0.94)	183429 (97.76)
Fracture	7270 (3.87)	484 (0.26)	179887 (95.87)
Support Device	105831 (56.4)	898 (0.48)	80912 (43.12)

Table 3.2: Number of Positive, Uncertain and Negative labels for each disease

The following table reports the numerical details about the dataset composition.

Dataset	**Frontal (%)**	**Lateral (%)**
Training	189116 (84.56)	32387 (14.48)
Validation	1911 (0.85)	0 (0.00)
Test	202 (0.09)	32 (0.01)

Table 3.3: Dataset structure

3.3 Preprocessing

In this section we describe the preprocessing steps that have been done on the data before submitting them to the learning algorithms.

Labels

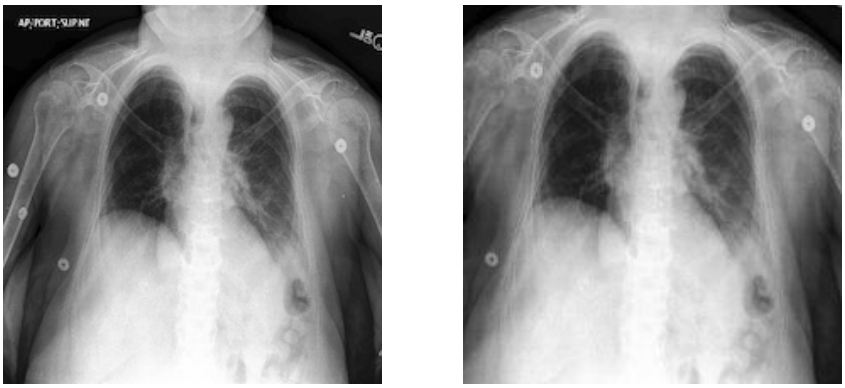
The author of the dataset provided, together with the images, two **.csv** files (one for the training set and one for the test set) mapping each CXR with its own set of labels. Moreover, each image has additional information about the gender and the age of the patient that, however, in this work have not been considered. We also decided to focus our attention on the Frontal Chest X-Rays only, discarding the Lateral ones, that were available only for a relatively small amount of patients.
These are the steps we have done in order to clean the **.csv** files:

- The columns **"Sex"**, **"Age"**, **"Frontal/Lateral"**, **"AP/PA"** have been removed.
- The *blank* values (i.e the unmentioned labels) have been assigned a value of 0.
- The uncertain labels, indicated with value -1, have been assigned a value that depends on the policy that have been used to deal with the uncertainties (we will give a more detailed explanation in section 4.1).

The final **.csv** files were composed by a column containing the image reference and 14 additional columns, one for each disease, filled with a value indicating whether a particular pathology is present, not present or uncertain.

Images

Most of the images were affected by irrelevant noise such as text or irregular borders. Since these areas might affect the learning performance, we tried to solve the problem by removing them. So the CXR images were first resized to 256×256 pixels and then a template matching algorithm has been applied in order to find a region of the image with size 224×224 containing a chest template. In many cases this approach successfully removed the irrelevant areas, as we can see in Figure 3.2. Moreover, the images have been converted to RGB in order to match the required input shape of the models and pixel values have been scaled in a range between 0 and 1. Finally, since the model we have used have been pre-trained on ImageNet [6], we have normalized all the images with respect to mean and standard deviation of that dataset.



Before After

Figure 3.2: Comparison of CXR before and after preprocessing

Data Format

When dealing with very large amount of data, using the proper data format could have a significant impact on the performance of the input pipeline and, consequently, on the training time of the model. In our work we have decided to use the TFRecord data format, that is Tensorflow's own binary storage format. TFRecord is extremely useful when dealing with large dataset that does not fit in RAM, because it allows to efficiently load and preprocess only the data that are required at the time. Each entry of our TFRecord file is composed by two elements:

- Image: serialized version of a CXR
- Labels: vector of 14 elements, each one associated to a particular pathology.

3.4 Related Works

Since it's introduction, there have been many works involving CheXpert. The first one has been released together with the dataset, in which the Stanford team developed a system able to classify and localize the different diseases exploiting a CNN. Moreover, they explored different approaches to deal with the uncertain labels and, finally, compared their results with that of 3 radiologist, exceeding their score in most of the pathologies.
The Stanford Machine Learning Group also set up a competition, called CheXpert [9], that allows people from all over the world to submit the result of their projects involving CheXpert. Among the different works, it worth mentioning that of Pham et al. [25], that reached the first position on the competition. The key point of their work is the investigation of novel approaches to deal with the uncertain labels, the use of Ensemble, and the exploitation of the dependency among the diseases to improve the quality of the prediction.

3.5 Summary

In this chapter we have introduced the dataset that we will use in our work, presenting its composition and its structure. We have also also talked about the preprocessing steps that we have done before submitting the data to the training algorithms and, finally, we mentioned the related works that have been done involving the same dataset. In the next chapter we will dive into the core of this work, presenting a formal definition of the problem and the solution we have developed to solve it.

Chapter 4

System Design

In this chapter we will present the approaches we have developed to solve the problem of classifying and localize the different diseases on a given Chest X-Ray. We will first talk about the approach we used to deal with uncertain labels, then we will present the models that we've developed to identify the different pathologies and we'll show the techniques that have been used to aggregate them. Finally, we will talk about the approach we've used to identify the region of the CXR affected by the disease.

4.1 Uncertain Labels

As already introduced in Chapter 3, the dataset contains a significant number of diseases that have been labelled as uncertain. The uncertainty could reflect both an unreliable diagnosis or an ambiguity in the report. In their work, Irvin et al. [12], proposed different policies to deal with them:

- U-Ignore: this is the simplest approach, that consists in ignoring all the uncertain labels during the training. This method, however, could produce biased models and also reduce the effective size of the dataset if the number of uncertain labels is significant.
- U-Zeros: this approach consists in assigning a 0 label to the uncertain one, thus considering the associated disease as not present.
- U-Ones: this is similar to the previous method but, instead of considering the uncertain labels as negative, they are considered as positive, thus they are assigned a label equal to 1.
- U-SelfTraining: with this method, the uncertain labels are first considered as unlabeled examples. Then a model is trained using the U-Ignore policy and, finally, it is used to make predictions in order to re-label each uncertain label.

- U-MultiClass: the last policy treats the u label as its own class. Thus, for each pathology, three different probabilities $\{p_0, p_1, p_u\}$ are carried out, each one representing the probability of belonging to, respectively, class 0, 1 or u.

Focusing our attention on U-Zeros and U-Ones policy, Pham et al. [25], in their work, put the evidence on a problem related to these two approaches: mapping all the uncertainty to either 0 or 1 will certainly result in a lot of wrong labels and, consequently, the model would be misguided during training. Thus, they used a technique called Label Smoothing Regularization (LSR), introduced by Szegedy et al. [34], that allows to exploit the large amount of uncertain labels but preventing the model from becoming over confident about the prediction of examples that could contain mislabeled data. Thus, the U-Ones policy has been modified by mapping each uncertain label to a random number close to 1. Formally, we have:

$$\bar{y}_k^{(i)} = \begin{cases} x, & \text{if } y_k^{(i)} = u \\ y_k^{(i)}, & \text{otherwise} \end{cases} \tag{4.1}$$

where $x \sim U(a_1, b_1)$ is a randomly distributed number between a_1 and b_1. Similarly, the U-Zeros policy can be modified, mapping the uncertain labels to a number $x \sim U(a_1, b_1)$ close to 0.

4.2 Convolutional Neural Network

In this section we talk about the first approach we've used, involving Artificial Neural Network. We start by describing the simplest model we've used and gradually we'll show how we have improved it.

4.2.1 Single CNN And Training Procedure

The first approach we have investigated is the one involving a Convolutional Neural Network trained with U-Ones policy and LSR (presented in the previous section). The uncertain labels, thus, have been mapped to a random number uniformly sampled between the interval $[0.55, 0.85]$.

Network Architecture

Instead of training a Neural Network from scratch, we have decided to rely on Transfer Learning, using a Neural Network previously trained on a different dataset and retraining it in order to adapt to our new task. The architecture we used is

the same already employed by Irvin et al. in their previous works [12] [26], called DenseNet121 and introduced for the first time in 2016 by Huang et al. [11]. The peculiarity of this architecture is the use of dense connections that prevent the problem of vanishing information as the depth of the network increases. This problem is addressed by connecting each layer directly to each other in order to ensure the maximum information flow between them. Thus a layer receives the input from all the preceding layers and passes its own output to all the subsequent ones, as we can see in the example in Figure 4.1

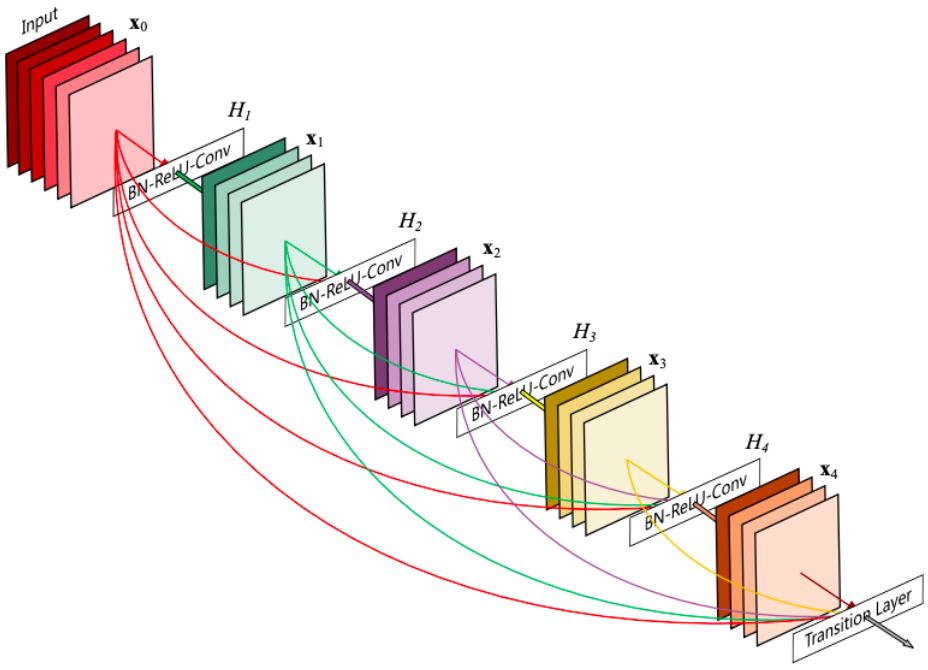


Figure 4.1: Densely Connected Neural Network

The weights of the network have been initialized with that of the same network trained on the ImageNet dataset. The Fully Connected layer has been removed and substituted by a Global Average Pooling followed by a new Fully Connected (FC) layer (note that the GAP is essential for the production of a Class Activation Map). The last layer produces a 14-dimensional output, after which we have applied an elementwise sigmoid function. If we call z_k the output corresponding to the k-th label, the result of applying a sigmoid is:

$$y_k = \frac{1}{1 + exp(-z_k)}, k = 1 \dots 14 \tag{4.2}$$

The sigmoid function allows to squeeze z_k in a range between $[0, 1]$, thus we can

give to y_k a probabilistic interpretation. Figure 4.2 summarizes the overall network architecture:

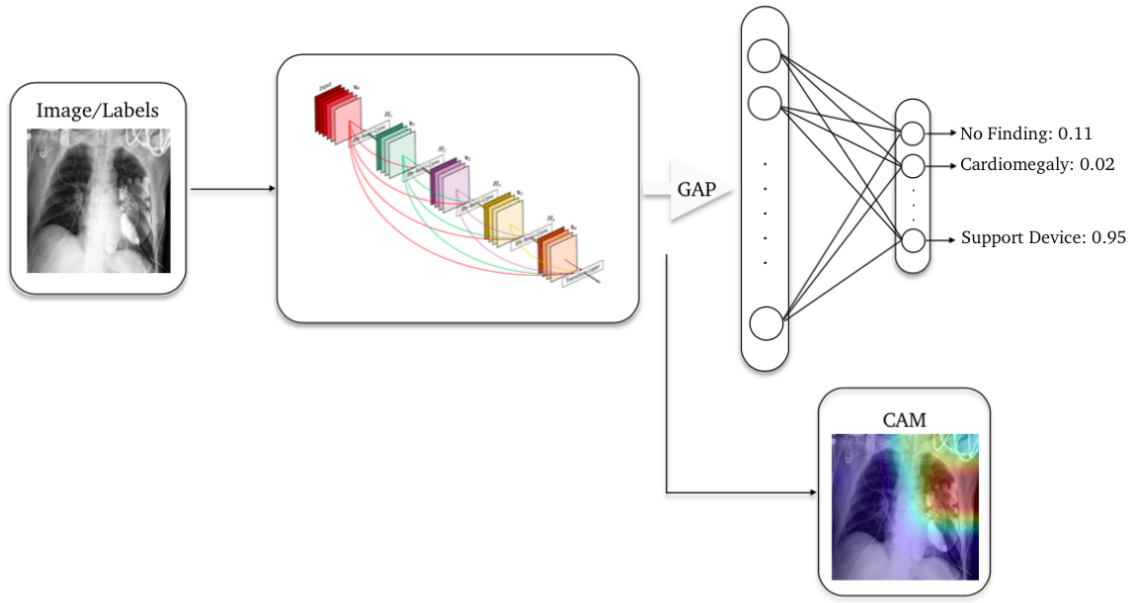


Figure 4.2: Network Architecture

Training

The network has been trained using Adam optimization algorithm [14] with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was initially set to $1e-4$ and then it has been gradually reduced by a factor of 10 at each epoch, allowing large weights changes at the beginning of the learning process and small changes towards the end. The number of epochs has been set to 10, but we also allowed the algorithm to early stop in case the optimization process show no more improvement on the validation set during the last 4 epochs. The objective function minimized by the optimizer is the binary cross entropy loss between the ground-truth labels and the predicted ones, who's formulation is the the following:

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{k=1}^{14} y_k^{(i)} \log \hat{y}_k^{(i)} + (1 - y_k^{(i)}) \log(1 - \hat{y}_k^{(i)}) \tag{4.3}$$

where $\boldsymbol{\theta}$ are the weights, $\hat{y}_k^{(i)}$ is the prediction associated to the i-th sample for class k and $y_k^{(i)}$ is its corresponding ground truth. Finally, after the data have been preprocessed following the steps described in section 3.3, they have been batched using a batch size of 64 and they have been shuffled in order for the data to be independent and identically distributed.

4.2.2 Conditional Training

As introduced in Section 3.2.1 the labels are organized according to a hierarchy (Figure 3.1). In the model we have defined in the previous section, however, we did not incorporated this knowledge, omitting an information that, instead, should be leveraged in order to help the model to better discriminate among the different diseases. The approach we are going to show has been proposed by Pham et al. in [25].

The proposed algorithm is divided in two phases. The first one is called *conditional training* and it's goal is to learn the dependency among parent and child labels. This can be done by training the classifier on data presenting only positive parent-level labels, teaching it to distinguish the lower-level ones (Figure 4.3). During the second phase, instead, the classifier will be fine tuned using the whole dataset, in this way it learns to classify also the parent-level labels, that could either be positive or negative. This is accomplished by freezing all the layers of the pretrained network except for the last fully connected one and training it again.

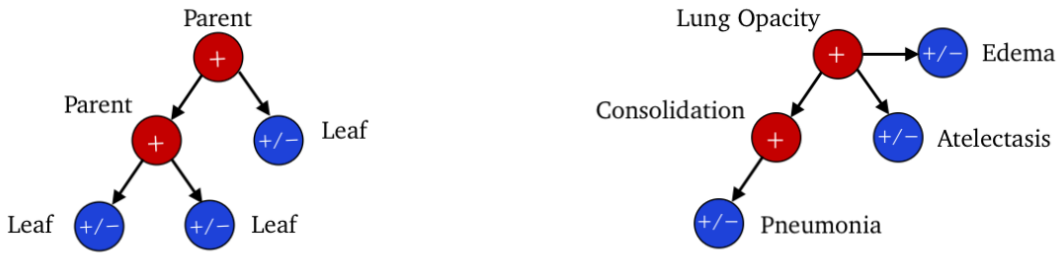


Figure 4.3: The left figure shows the key idea behind conditional training. In the right one there is an example, in which the classifier is trained on samples containing only positive instances of *Lung Opacity* and *Consolidation* in order to classify the child labels *Edema*, *Atelectasis* and *Pneumonia*.

Using this training strategy, the output of the network can be viewed as the conditional probability that a label is positive giving its parent label being positive. Our goal, however, is to predict the unconditional probability. In order to do that we can rely on Bayes Theorem, calculating the probability of each label being positive by multiplying each conditional probability produced by network along the path from the root to the current node. For example, considering the tree depicted in Figure 4.4, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are parent labels, while $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ are leaf:

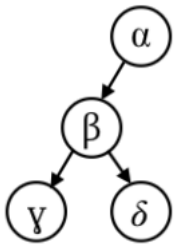


Figure 4.4: Example of tree presenting four diseases

The network provides the conditional probabilities $p(\alpha)$, $p(\beta|\alpha)$, $p(\gamma|\beta)$, $p(\delta|\beta)$. The unconditional predictions for labels $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$, instead, are given by:

$$\begin{aligned} p(\boldsymbol{\gamma}) &= p(\boldsymbol{\alpha})p(\boldsymbol{\beta}|\boldsymbol{\alpha})p(\boldsymbol{\gamma}|\boldsymbol{\beta}) \\ p(\boldsymbol{\delta}) &= p(\boldsymbol{\alpha})p(\boldsymbol{\beta}|\boldsymbol{\alpha})p(\boldsymbol{\delta}|\boldsymbol{\beta}) \end{aligned} \tag{4.4}$$

Note that this approach ensures that the probability of presence of a disease for a child label is always smaller than that of its parent.

4.2.3 Neural Network Ensemble

In Chapter 2 we have talked about Ensemble, a widely used Machine Learning technique who's aim is to gather the predictions coming from different classifiers and combine them in order to enhance the quality of the predictions.

In a multilabel classification setting it's hard to obtain high results for each target using a single classifier and, often, the score for each label varies with the choice of the architecture. So we decided to rely on Embedding, trying to combine several weak classifiers into a stronger one. Again, instead of building our own architecture from scratch, we used some of the most popular pretrained network emplyoed for image classification tasks. The list of the network we have used, along with some information about their structure, is shown in Table 4.1

Name	#Parameters	Depth	Size	Year
DenseNet121	7M	121	33MB	2016 [11]
DenseNet169	12,5M	169	57MB	2016 [11]
DenseNet201	18M	201	80MB	2016 [11]
InceptionResNetV2	54M	572	215MB	2016 [33]
Xception	21M	126	88MB	2016 [5]
VGG16	15M	23	528MB	2014 [31]
VGG19	20M	26	549MB	2014 [31]

Table 4.1: List of pretrained CNN used

All the Networks have been trained following the same approach used for the training of DenseNet121 (described in Section 4.2.1). At the end of the training process, thus, we had 7 classifiers, each one carrying out a different set of predictions. We will now talk about the approaches we investigated to combine them.

Average

The first (and simplest) approach we have used consist in averaging the predictions made by the classifiers. If we indicate with $y_i(\mathbf{x})$ the 14-dimensional output of classifier i, then the final prediction $\tilde{y}(\mathbf{x})$, computed using the predictions coming from all the N classifiers, is given by:

$$\tilde{y}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{4.5}$$

The main drawback of this approach, however, is that it gives the same importance to all the models, without taking care of the fact that some classifiers may outperform other in classifying some diseases and, thus, their opinion should have a higher weight in the process of carrying out the final prediction.

Weighted Average

We tried to overcome the problem that arose using a simple average aggregation by using a different weight for each model. The weights have been assigned using an heuristic we developed that is based on the concept of Entropy. In Information Theory the Entropy is a quantity associated to a random variable that capture the level of uncertainty related to its possible outcome. For example, consider tossing a coin: in the case in which the coin is fair (i.e head and tail have both the same probability to occur), we are very insecure about the result, thus in this case the entropy is maximum. If, instead, the coin is not fair and one result has a higher probability to occur than the other, the result will be less uncertain and, consequently,

the entropy will be lower. We used this quantity to measure the level of uncertainty of each outcome. Consider, for example, the prediction p_i for label i, taken out by a given model. We can compute the entropy of the predicions as:

$$H(p_i) = -p_i \log_2 p_i - (1 - p_i) \log_2 (1 - p_i) \tag{4.6}$$

Note that, if $H(p_i)$ captures the information related to the uncertainty of a prediction, $(1 - H(p_i))$, instead, gives us an insight related to how safe the model is about it.
Now that we have introduced the concept of entropy we can describe how we used it in our task. Consider a CXR given as input to our $N = 7$ predictors. The output associated to the $L = 14$ labels can be represented as a matrix:

$$P = \begin{bmatrix} p_{11} & \dots & p_{1L} \\ \vdots & \ddots & \vdots \\ p_{N1} & \dots & p_{NL} \end{bmatrix} \tag{4.7}$$

where p_{ij} is the predictions carried out by i-th model for label j. Using equation 4.6 we can also compute a matrix of weigths:

$$H = \begin{bmatrix} h_{11} & \dots & h_{1L} \\ \vdots & \ddots & \vdots \\ h_{N1} & \dots & h_{NL} \end{bmatrix} \tag{4.8}$$

Now, the final prediction y_i for label i can be calculated as a weighted average of all the outputs, where a weight captures the certainty of the model with respect its own prediction. Here's the mathematical formulation:

$$y_i = \sum_{k=1}^{N} (1 - h_{ki})\, p_{ki} \tag{4.9}$$

Stacking

The last aggregation approach we investigated is the one involving Stacking, a technique already presented in Section 2.3. The idea is to let an algorithm learn the best way to combine the predictions instead of doing it manually. We have used the predictions generated over our validation set as input to train a meta-learner, who's aim is that of aggregating the outputs in order to generate the final prediction. Note that in order to train the meta-learner it's very important to use data that have not been employed during the training of the base classifiers, otherwise we would incur in overfitting. The

predictions coming from all the networks have been stacked together, thus the size of a single example was constituded by $L \times N$ data points, where L is the number of labels and N the number of predictors. As meta-learner we used a Random Forest algorithm, whose hyperparameters have been tuned using a Grid Search approach.

4.3 Embedding

The inherent ability of CNNs to deal with images made them a standard for the task of CXRs analysis. In this section, however, we try to investigate other approaches that can be used to classify the pathologies. For many algorithms, however, it's almost impossible to learn the discriminative features directly from the image, unless it has a very small size. It would be useful, then, to have a small representation of the CXR that, however, is able to capture all the meaningful informations nedeed by the algorithm. Neural Networks can provide us such a representation by using a technique called Embedding, presented in section 2.2.3. When the input image is processed by the network, indeed, it pass through a sequence of layers who's aim is to extract the meaningful features that are subsequently used to feed the fully connected classifier. During this process, moreover, the image is downsampled by the pooling layer, thus its spatial size is reduced as it flows through the layers. So, if we cut the network at a certain point, the output we observe is a representation of the original input embedded into a lower dimensional space. In our work we have extracted the embedding using the Neural Networks we trained described in Section 4.2.3 (note that each model provided us a different embedding). All the Networks have been cut at the same point, that is immediately after the Global Average Pooling layer, producing a one-dimensional vector. Table 4.2 provides an insight about the shape of the embedding for each model:

Model Name	**Embedding Shape**
DenseNet121	(1,1024)
DenseNet169	(1,1664)
DenseNet201	(1,1920)
InceptionResNetV2	(1,1536)
Xception	(1,2048)
VGG16	(1,512)
VGG19	(1,512)

Table 4.2: Embedding Shape

4.3.1 Random Forest

Once we've extracted the embeddings, they can be used as input for other models. The first approach we have explored is that involving Random Forest. As introduced in Chapter 1, Random Forest is a learning algorithm that combines a group of decision trees and aggregates their output. Since the embeddings we had were different from each other, we decided to train a different Random Forest for each one of them. Using the validation set, we have optimized the hyperparameters by means of a Grid Search algorithm, that is an exhaustive search through a manually specified subset of values. The following hyperparameters have been taken into consideration:

- **Number of estimators**: the number of tree in the forest. Due to computational reason it has been set 200 for all our models.
- **Max features**: number of features to consider when looking for the best split (for each model we set it equal to the square root of the total number of variables).
- **Max depth**: maximum depth of the trees. The deeper the tree, the more precise it is, but this could lead to overfitting.
- **Min sample split**: minimum number of samples required to split an internal node.
- **Min sample leaf**: minimum number of samples required to be at a leaf node.

Table 4.3 contains a summary of hyperparameters we used for each model.

Model Name	Max Depth	Min Sample Split	Min Sample Leaf
DenseNet121	15	2	10
DenseNet169	15	2	10
DenseNet201	30	10	10
InceptionResNetV2	30	10	1
Xception	30	10	1
VGG16	5	2	10
VGG19	15	50	1

Table 4.3: Random Forest Hyperparameters

Once the models have been trained, we combined their predictions using the same approaches discussed in the previous section (simple average, weighted average with entropy and stacking).

4.3.2 XGBoost

The last approach we tried is the one involving XGBoost (eXtreme Gradient Boosting) [4], that is an efficient implementation of Gradient Boosting, introduced in section 2.5. As already mentioned, this algorithm iteratively improves the predictions, adjusting them by making a step in the direction minimizing the loss function, which is given by the gradients. We trained a different classifier for each one of the embedding we extracted from our Neural Networks and we combined their predictions using the same approaches described in the previous section. In order to prevent overfitting, we constrained the structure of the trees, setting the maximum depth to 3 and then we trained each models for 50 rounds.

4.4 Class Activation Map

As previously stated, the goal of this work is double: predicting the probability, for a given pathology, to be present on a CXR and also localize it. Up to this moment we focused the attention on the first task, thus we want now to talk about the second one. The Neural Networks we trained allowed us to extract the Class Activation Map, that is the area that the model predicts to be the most indicative of each observation.The activation maps have been generated following the methodology proposed in [27] and illustrated in Section 2.2.2: given a trained CNN, the CAM can be computed by making a weighted sum of the feature maps extracted by the last convolutional layer. Since we trained many models, we were able to compute multiple Class Activation Maps and, as already made during the classification task, combine them in order to improve the quality. However it would have made no sense to aggregate indiscriminately all of them, because we would risk to combine a map produced by a model predicting the absence of a pathology with one coming from a model telling the opposite. Thus we have decided to merge only the heatmaps coming from models having the same prediction over the presence or absence of the disease.

4.4.1 Bounding Box Generation

Once we have generated the CAM, we used them to automatically generate a bounding box surrounding the area associated to the pathology. In order to do that, we first used the Class Activation Map as a mask to determine the salient area of the image. In order to do that, we have converted each pixel of the CAM to a binary value, using

the following rule:

$$b(x,y) = \begin{cases} 1, & \text{if CAM(x,y)} \geq threshold \\ 0, & \text{if CAM(x,y)} < threshold \end{cases} \tag{4.10}$$

where $b(x,y)$ is the intensity value of a pixel at position (x,y). The threshold, instead, has been computed as:

$$threshold = \alpha \times max(\text{CAM}), \alpha \in [0,1] \tag{4.11}$$

Once we got the mask, it has been overlapped to the original image I, and the saliency region $p(x,y)$ has been extracted by making a pointwise product:

$$p(x,y) = I(x,y) \cdot b(x,y) \tag{4.12}$$

Once we got that image, we were able to extract the coordinates of its contours and, using them, we draw the bounding box over the original image. Figure 4.5 shows the full process used to automatically extract the bounding box. The first figure contains the original CXR, the second one shows the generated CAM, the third one represents the saliency region $p(x,y)$ extracted using the heatmap and, finally, the last one shows the automatically generated bounding box.

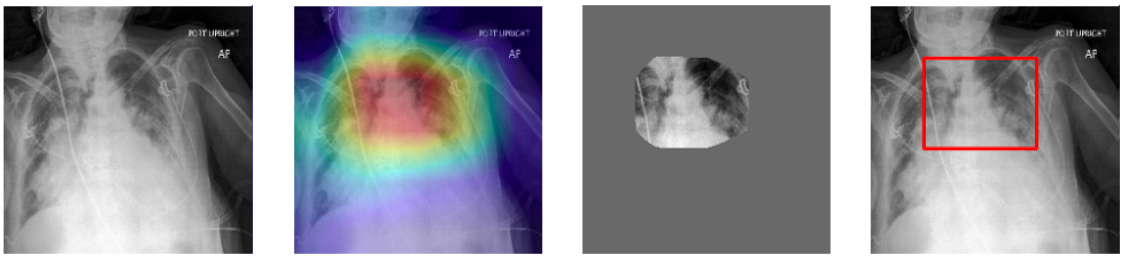


Figure 4.5: Bounding Box Generation

4.5 Summary

In this chapter we have shown all the techniques we have used to solve the problems of predicting and localizing the different diseases. We started showing how to better deal with the uncertain labels, then we have built the first model that, gradually, we have modified, trying to improve the overall predictions perfomances. Finally we have shown the ability of the trained models to localize the pathologies on the CXR and how this localization ability can be used to automatically generate a bounding box sorrounding the region interested by the pathology. In the next chapter we will talk

about the metrics we used to evaluate our models and we'll exhibit and discuss the results we achieved.

Chapter 5

Results

In this chapter we will first introduce the methodology we used to evaluate our models and then we will show and discuss the results we have obtained both in classifying the diseases and in localizing them.

5.1 Evaluation Criteria

As introduced in Chapter 3, the dataset already provides a subset of 202 frontal images that can be used to asses the performances of the models. We decided, in our work, to evaluate our models using the same subset, in order to be able to compare our results with that carried out by related works. The performance, however, have not been evaluated over the entire set of pathologies but, instead, only 5 of them (*Atelectasis*, *Cardiomegaly*, *Consolidation*, *Edema*, *Pleural Effusion*) have been taken into consideration, based on their clinical significance and presence in the dataset.

AUROC

Area Under Receiving Operating Characteristic (AUROC) is the summary metric we used to assess the goodness of our classifiers. In order to define it, we have to first introduce some quantities that are largely used to measure the performance of a classifier:

- True Positive (TP): outcome *correctly* classified as *positive*
- True Negative (TN): outcome *correctly* classified as *negative*
- False Positive (FP): outcome *incorrectly* classified as *positive*
- False Negative (FN): outcome *incorrectly* classified as *negative*

We can now introduce the notion of Receiving Operating Characteristic, that is a curve plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). TPR, also known as sensitivity or recall, measures the proportion of actual positive that are identified as such, and is computed as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.1}$$

FPR, instead, measures the probability of a false alarm, i.e the number of negative samples incorrectly classified as positive, and is given by:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{5.2}$$

In order to compute these quantities, however, the predictions, that are expressed as probabilities, must be converted into binary decisions, in other words a value telling whether a given sample is classified as positive or negative. This can be done by defining a value of the probability, called decision threshold, above which the samples are classified as positive while, otherwise, they are assigned to the negative class. Once a threshold value has been selected, the performance of a classifier can be represented as a point in the ROC space. Consider, for example, Figure 5.1:

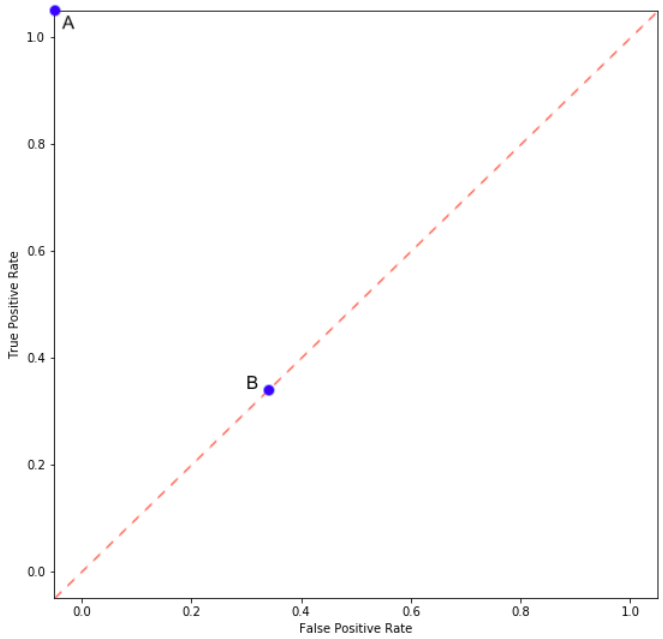


Figure 5.1: ROC Example

Point A=(0,1) represents the ideal case, in which all the samples are assigned to the correct class. Point B, instead, lies on the so called *line of no discrimination*, whose performance is the same of a random guessing. Note that the line of no discrimination splits the ROC space into two parts: points above the line represents good classification results, while points below represents bad results (worse than random). A way to asses the overall performance of a predictor is that of drawing a curve on the ROC plane by plotting its score for every possible threshold value used to discriminate between positive and negative class. Then we can finally calculate the area under the curve, obtaining the AUROC value (also called AUC). For what it concern the medical field,according to [18], an AUC of 0.5 suggests no discrimination, 0.7 to 0.8 is considered acceptable, 0.8 to 0.9 is considered excellent, and more than 0.9 is considered outstanding.

5.2 Experimental Design

The first experiment we conducted involved a single CNN, more specifically a DenseNet121. Using U-Ones as policy for dealing with uncertain labels, we trained it for 5 epochs using a training set composed by 189116 samples. The learning rate was initially set to 1e-4 and then reduced by a factor of 10 after each epoch. As objective function we used the binary cross-entropy. Then we have repeated the experiment but using U-Ones+LSR as policy for dealing with uncertain labels. Next step was that of exploiting the diseases taxonomy, using Conditional Training. During the first phase we used a training set composed by 90954 samples and we trained the DenseNet121 network end to end, using the same set up of the previous experiments. Then we froze all the layers except the last fully connected one and we retrain the network using all the 189116 samples for 5 epochs. The same procedure has been repeated for each architecture indicated in Table 4.1, obtaining the 7 different models that we then combined exploiting different ensemble techniques. Moreover, the networks have been used to generate the embeddings, extracted from the Global Average Pooling layer and then used to train both Random Forest and XGBoost. The hyperparameters have been tuned using a validation set composed by 1911 samples.

5.3 Experiment Results

We will now present and discuss the results we obtained in our experiments, using the models we have introduces in the previous chapter. We will compare each other, showing, for each one, the benefits and the downsides.

5.3.1 Single CNN

Our first objective was to check the effectiveness of Label Smooth Regularization and Conditional Training. Starting from a simple model, that we used as baseline, we then implemented the two strategies, checking whether they were able to bring some improvements or not. As baseline we used a DenseNet121 architecture (the same used in [12] [25]), trained with U-Ones policy for dealing with uncertain labels. As we can see in Figure 5.2, this approach was able to achieve a mean AUROC across the five diseases of 0.865 which, according to [18], is already a good result. The network, however, showed difficulties in recognizing the cases of Atelectasis and Cardiomegaly, while, instead, was able to identify pretty well the CXRs containing Edema and Pleural Effusion that, is worth noting, are the most balanced diseases among those we are considering (Table 3.2).

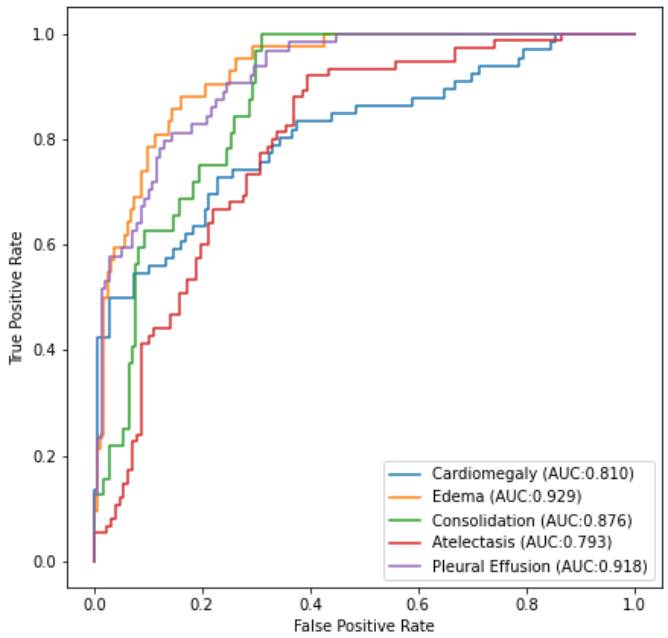


Figure 5.2: ROC curve of DenseNet121 architecture trained with U-Ones policy

Using the Label Smooth Regularization, together with the U-Ones policy, led to an improvement in recognizing all the pathologies except for Consolidation, where the previous approach outperformed the current one (Figure 5.3).

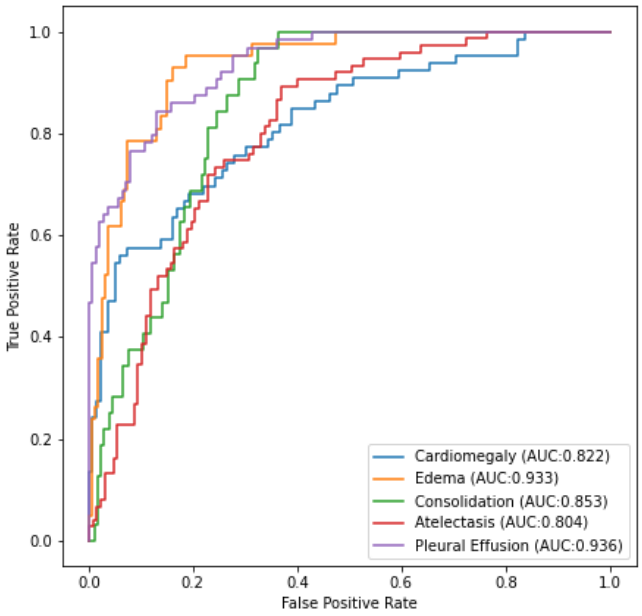


Figure 5.3: ROC curve of DenseNet121 trained with U-Ones policy + Label Smooth Regularization.

The use of Conditional Training, instead, gave a big improvement in identifying the cases of Atelectasis and Consolidation (Figure 5.4), with the downside of slightly worsening the performance over the other diseases. The mean AUROC, however, is the best obtained so far (0.876), confirming the hypothesis that using additional side information, such as dependencies among the pathologies, can help improving the overall performance of the network.

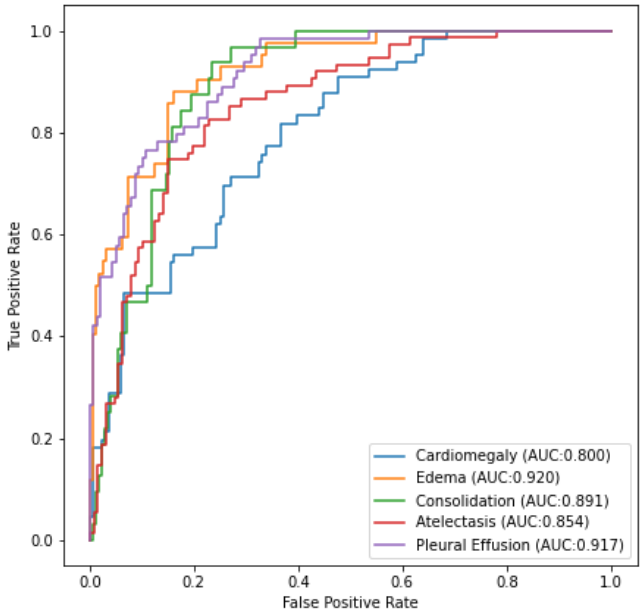


Figure 5.4: ROC Curve of DenseNet121 trained with U-Ones policy + Label Smooth Regularization + Conditional Training.

Table 5.1 shows the comparison of the three different approaches we've investigated using a single CNN.

Method	**Atelectasis**	**Cardiomegaly**	**Consolidation**	**Edema**	**P. Effusion**	**Mean**
U-Ones	0.793	0.810	0.876	0.929	0.918	0.865
U-Ones+LSR	0.804	**0.822**	0.853	**0.933**	**0.936**	0.869
U-Ones+LSR+CT	**0.854**	0.800	**0.891**	0.920	0.917	**0.876**

Table 5.1: AUROC values obtained using DenseNet121 as architecture and the approaches indicated in the **Method** column. For each label, the highest AUROC is boldfaced.

5.3.2 CNN Ensemble

Table 5.2 shows the results achieved by the other CNN architectures trained using U-Ones+LSR+CT policy. The mean AUROCs are very similar among each other and they all present analogous behaviour in distinguishing the individual diseases: the greatest difficulties are encountered while recognizing the cases of Cardiomegaly, while they all achieve the best results identifying Edema and Pleural Effusion. The results we obtained confirm the fact that there is no an architecture outperforming all the others on each pathology. Consider, for example, the VGG19 network: it achieved

the best result in recognizing Consolidation and Pleural Effusion but, instead, its performance on Cardiomegaly is the worst.

Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
DenseNet121	**0.854**	**0.800**	0.891	0.920	0.917	**0.876**
DenseNet169	0.850	0.795	0.882	**0.936**	0.915	**0.876**
DenseNet201	0.834	0.791	0.881	0.917	0.925	0.870
InceptionResNetV2	0.816	0.784	0.897	0.925	0.919	0.869
Xception	0.841	0.770	0.880	0.909	0.924	0.865
VGG16	0.843	0.772	0.898	0.932	0.919	0.873
VGG19	0.843	0.769	**0.900**	0.927	**0.933**	0.874

Table 5.2: AUROC values obtained with the CNNs indicated in the **Model** column.

Once we trained the different networks, we tried to improve the overal diagnostic capability by combining them. Figures 5.5 and 5.6 show, respectively, the results of applying simple Average and Entropy Weighted Average as aggregation methods. As we can see, both the approaches helped improving the model's ability to recognize each of the five pathologies. More specifically, as we can see from Table 5.3, making an entropy weighted average that took care of each model's accuracy turned out to be a successful aggregation strategy, leading to a mean AUROC of 0.889, a result that slightly outperformed the simple average approach.

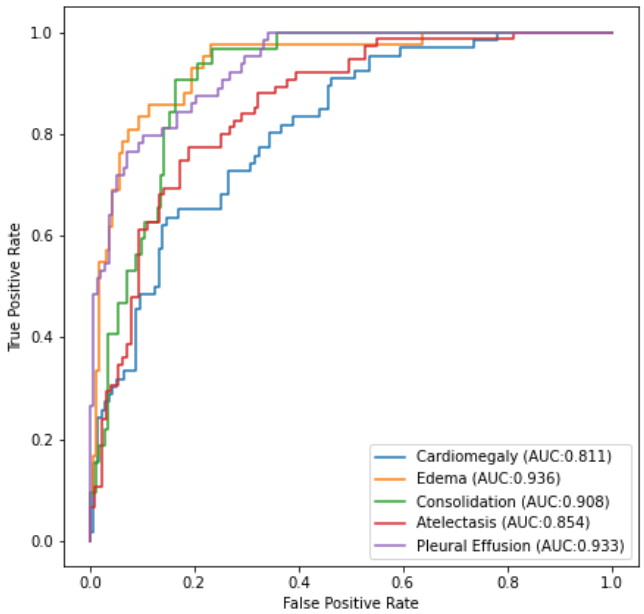


Figure 5.5: ROC curve of the CNNs ensemble computed using simple average.

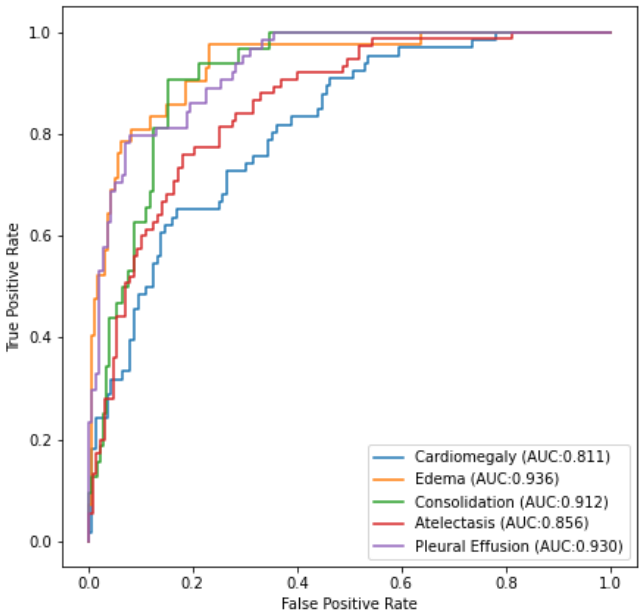


Figure 5.6: ROC curve of the CNNs ensemble model computed using average weighted by predictions' entropy.

An aggregation approach based on Stacking, however, didn't boost the overall performance, leading to a significant improvement only in Pleural Effusion recognition.

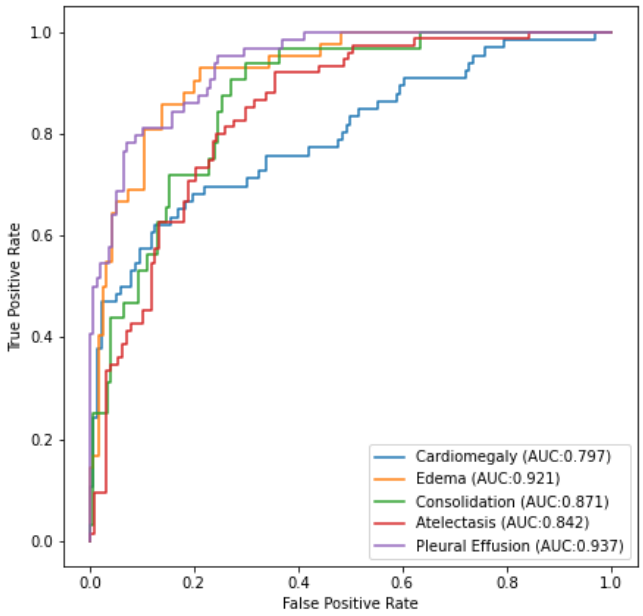


Figure 5.7: ROC curve of the CNNs ensemble model computed using stacking.

Table 5.3 reports the AUROC obtained by the three ensemble policies, compared to the best results achieved using a single CNN. We can clearly see that aggregating the predictions led to a significant improvement. In particular, making an entropy

weighted average that took care of each model's accuracy turned out to be a successful aggregation strategy, leading to a mean AUROC of 0.889, a result that slightly outperformed the simple average approach.

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
Single CNN	0.854	0.800	0.900	**0.936**	0.933	0.885
Average Ensemble	0.854	**0.811**	0.908	**0.936**	0.933	0.888
Entropy Weighted AVG	**0.856**	**0.811**	**0.912**	**0.936**	0.930	**0.889**
Stacking	0.842	0.797	0.871	0.921	**0.937**	0.873

Table 5.3: AUROC values obtained using different Ensemble policies, compared to the best results achieved by a single CNN (reported in the first row). For each label, the highest AUROC is boldfaced.

5.3.3 Random Forest with Embedding

Once we trained the CNNs, we used them to extract the embeddings. The first algorithm we used exploting them is Random Forest, whose results are shown in the Table 5.4.

Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
DenseNet121	0.851	*0.818*	0.885	0.915	***0.945***	*0.883*
DenseNet169	*0.855*	*0.814*	*0.893*	**0.922**	*0.933*	***0.884***
DenseNet201	*0.863*	*0.814*	0.878	***0.922***	*0.936*	*0.882*
InceptionResNetV2	*0.830*	0.779	*0.898*	0.918	*0.933*	*0.872*
Xception	0.831	*0.810*	*0.907*	*0.913*	*0.932*	*0.879*
VGG16	*0.858*	***0.822***	***0.913***	0.886	0.917	*0.879*
VGG19	***0.873***	*0.798*	0.895	0.892	0.917	*0.875*

Table 5.4: AUROC values obtained with Random Forest trained using the embedding extracted from the CNN indicated in the **Model** column. For each label, the highest AUROC is boldfaced. A score written in italics means that the RF, on that label, outperformed the result achieved by its corresponding CNN, reported in Table 5.2

As we can see, each of the single models we trained achieved, on average, a better result than its corresponding one (shown in Table 5.2), confirming the fact that the embeddings are actually able to capture the relevant information needed to distinguish the different diseases.

As in the previous case, we combined the predictions carried out by the different models, investigating the same approaches used before. Figures 5.8 and 5.9 show the results obtained using, respectively, Average and Entropy Weighted Average as ensemble policy.

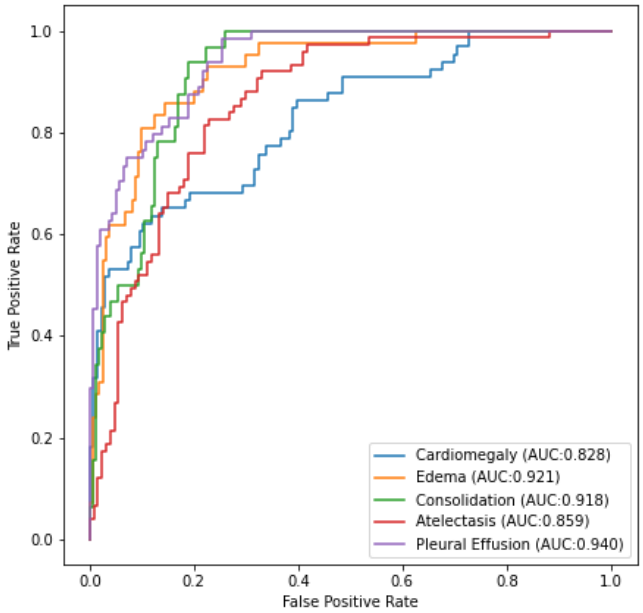


Figure 5.8: ROC curve of the RF ensemble model computed using simple average.

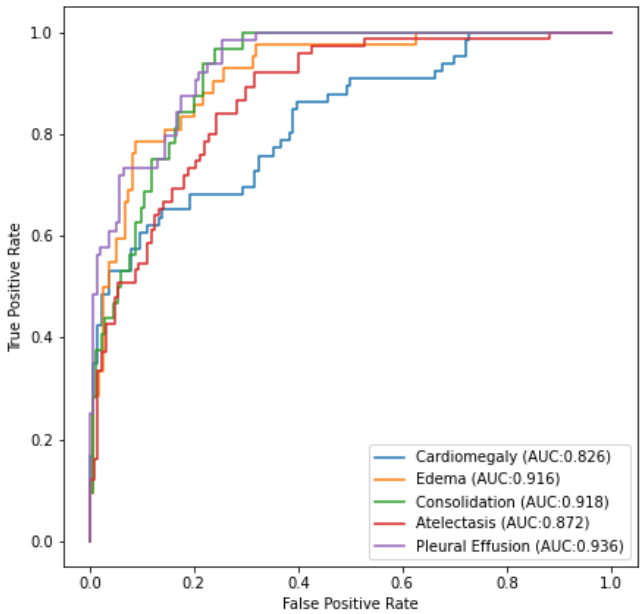


Figure 5.9: ROC curve of the RF ensemble model computed using average weighted by predictions' entropy.

They both brought to a significant improvement, with the latter achieving a mean AUROC of 0.897 (Table 5.5), the highest result among those we have seen so far. Using stacking as aggregation method, instead, didn't help improving the diagnosis. On the contrary, this approach drastically reduced the performance on Cardiomegaly, leading to one of the worst score on that label.

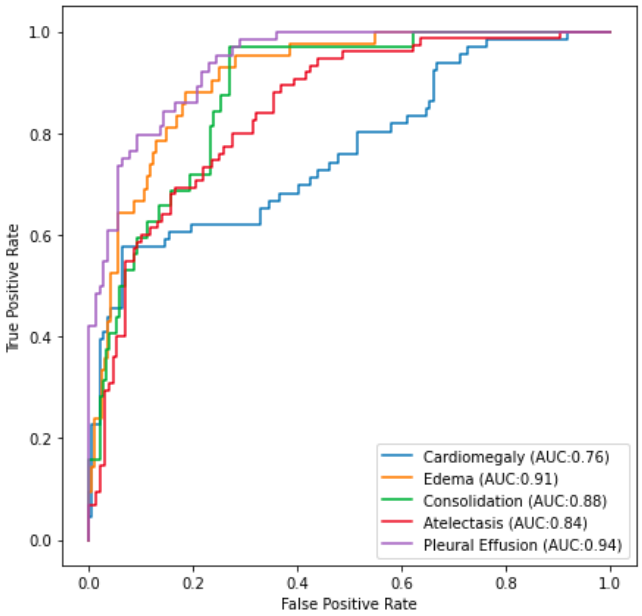


Figure 5.10: ROC curve of the RF ensemble model computed using stacking.

We can see the results of the three ensemble methods in Table 5.5, compared to the best result achieved by a single Random Forest (the one obtained by using the embedding extracted by the DenseNet169 architecture).

Method	**Atelectasis**	**Cardiomegaly**	**Consolidation**	**Edema**	**P. Effusion**	**Mean**
Single RF	0.855	0.814	0.893	0.922	0.933	0.884
Average	0.859	**0.828**	**0.918**	**0.921**	**0.940**	0.893
Entropy Weighted AVG	**0.872**	0.826	**0.918**	0.916	0.936	**0.897**
Stacking	0.840	0.761	0.883	0.908	0.937	0.866

Table 5.5: AUROC values obtained using different Ensemble policies combining Random Forest models trained using the embeddings. For each label, the highest AUROC is boldfaced.

Table 5.6, instead, shows the comparison between the best result achieved using Neural Networks with respect to that obtained by the embeddings. As we can clearly see, the latter is able to outperform the ensemble of CNNs in the recognition of all the diseases except for Edema.

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
CNNs	0.856	0.811	0.912	**0.936**	0.930	0.889
RF Embeddding	**0.872**	**0.826**	**0.918**	0.916	**0.936**	**0.897**

Table 5.6: AUROC values comparison between CNNs and RFs trained using embedding. For each label, the highest AUROC is boldfaced.

5.3.4 XGBoost

The embeddings have been used to trained another set of models, based on XGBoost, whose results are reported in Table 5.6. All of them, except for DenseNet201, achieved results similar to those obtained with the CNNs. Random Forest, thus, seems to be more capable of exploiting the embedding to classify the CXR.

Table 5.7: AUROC values obtained with XGBoost trained using the embedding extracted from the CNN indicated in the **Model** column.

Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
DenseNet121	0.803	*0.837*	0.837	***0.944***	*0.935*	0.871
DenseNet169	0.812	*0.829*	0.866	0.916	*0.923*	0.869
DenseNet201	**0.824**	***0.867***	0.876	*0.920*	***0.938***	***0.885***
InceptionResNetV2	*0.820*	*0.792*	***0.911***	0.911	*0.922*	*0.871*
Xception	0.803	*0.804*	*0.901*	0.899	0.923	*0.866*
VGG16	*0.800*	0.840	0.849	0.922	*0.927*	0.868
VGG19	0.811	*0.819*	0.832	0.921	0.922	0.861

Combining the XGBoost models, however, the performance increased significantly with respect to using the single models alone (Figure 5.11 and 5.12). In particular, differently from the previous approaches, by combining the XGBoost predictions the system is able to achieve good results also on Cardiomegaly that, according to the previous results, was one of the most difficult pathology to identify.

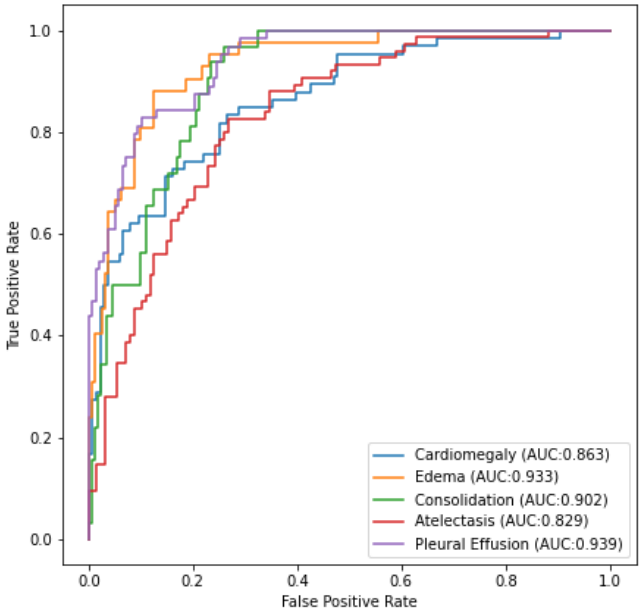


Figure 5.11: ROC curve of the XGBoost ensemble model computed using simple average.

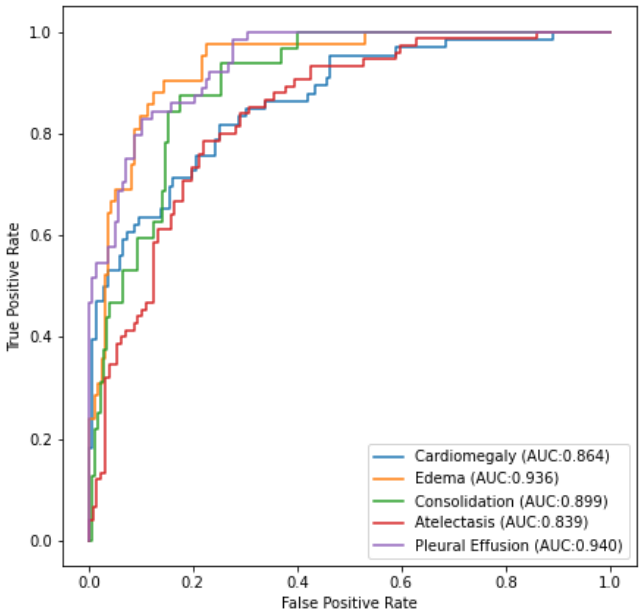


Figure 5.12: ROC curve of the XGBoost ensemble model computed using average weighted by predictions' entropy.

The results obtained after the aggregation are summarized in the Table 5.8, along with the best result obtained using CNN. Again, the use of the embedding is able to outperform the results achieved using the full images.

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
CNNs	**0.856**	0.811	**0.912**	**0.936**	0.930	0.889
XGB Average	0.829	0.863	0.902	0.933	0.939	0.893
XGB Entropy Weighted AVG	0.839	**0.864**	0.899	**0.936**	**0.940**	**0.896**

Table 5.8: AUROC values obtained using different Ensemble policies combining XGBoost models trained using the embeddings. For each label, the highest AUROC is boldfaced.

5.3.5 Final Results

The final predictions have been obtained by making a simple average of the three different ensembles (CNN, RF and XGBoost), each one computed using a weighted average ensemble policy and trained using U-Ones+LSR+CT. In Figure 5.13 are reported the ROCs curve obtained, one for each pathology.

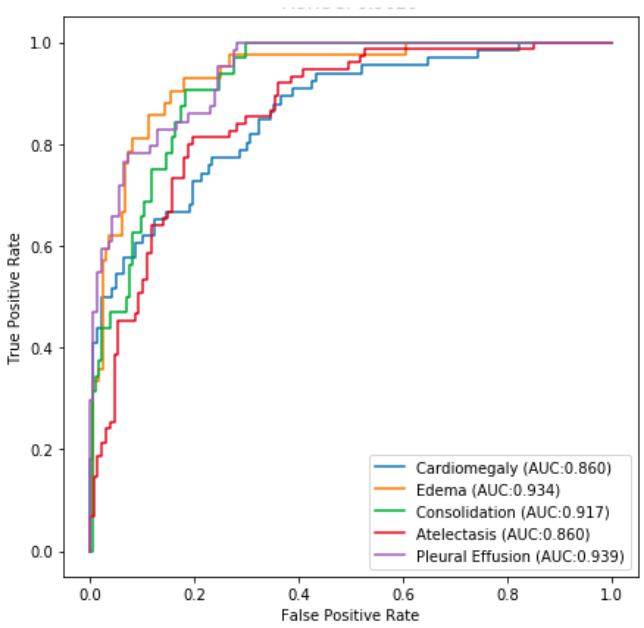


Figure 5.13: ROC curve of the final model, obtained by aggregating the ensembling of CNN, RF and XGBoost.

Table 5.9, finally, reports a summary of all the results we obtained so far, showing the model from which we started (DenseNet121+U-Ones), the best models obtained using CNNs, RFs and XGBoost (trained using U-Ones+LSR+CT), their ensemble (obtained using Entropy Weighted Average as ensemble policy) and the final model, that achieved a mean AUROC of 0.902. The biggest improvement, with respect to the starting point, is the one involving Atelectasis and Cardiomegaly, that are the diseases on which the initial model performed worse.

Method	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
DenseNet121 + U-Ones	0.793	0.810	0.876	0.929	0.918	0.865
CNN (DenseNet121)	0.854	0.800	0.891	0.920	0.917	0.876
RF (DenseNet169)	0.855	0.814	0.893	0.922	0.933	0.884
XGB (DenseNet201)	0.824	**0.867**	0.876	0.920	0.938	0.885
CNN Ensemble	0.855	0.811	0.912	0.936	0.930	0.889
RF Ensemble	**0.872**	0.826	**0.918**	0.916	0.936	0.897
XGB Ensemble	0.839	0.864	0.899	**0.936**	**0.940**	0.896
Final Ensemble	0.860	0.860	0.917	0.934	0.939	**0.902**

Table 5.9: AUROC comparison of the different approaches investigated in this work. Note that, for the single models, we reported only the one achieving the highest mean result (the name of the corresponding architecture is reported between brackets).

Our final model was able to outperform, on average, the one proposed by the author of the Chexpert dataset [12], whose results are shown in the Table 5.10

Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
Chexpert	0.821	0.854	**0.937**	0.928	0.936	0.895
Ours	**0.860**	**0.860**	0.917	**0.934**	**0.939**	**0.902**

Table 5.10: AUROC comparison with Chexpert author's model (for simplicity we only report, among the results they obtained, the one with the highest mean AUROC).

We weren't able, however, to outperform the system developed by Pham et al. [25], whose results are reported in Table 5.11. Nevertheless we achieved good results using different approaches and, in particular, we show that using the embedding is possible to develop a CXR diseases classifier using algorithm that are computationally much less expensive than CNNs.

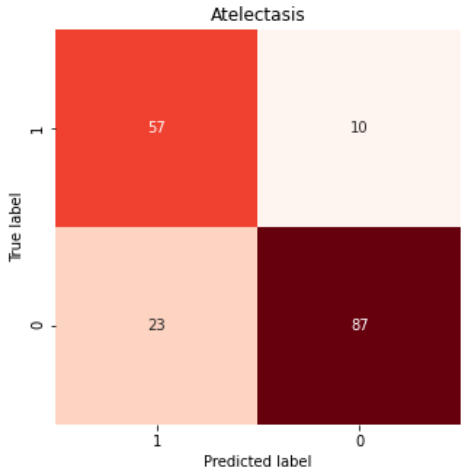
Model	Atelectasis	Cardiomegaly	Consolidation	Edema	P. Effusion	Mean
Ours	0.860	0.860	0.917	0.934	0.939	0.902
Pham et al.	**0.909**	**0.910**	**0.957**	**0.958**	**0.964**	**0.940**

Table 5.11: AUROC comparison with Pham et al. model.

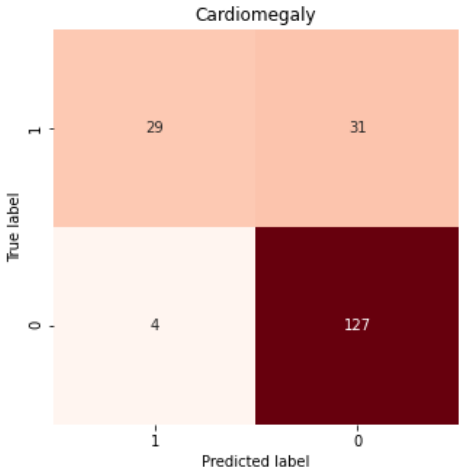
Confusion Matrices

Figure 5.14 shows the confusion matrices for the five diseases, a more intuitive way to display the output of our model. For each pathology we used a different decision threshold. They have been tuned using the validation set, by taking, for each disease, the average output activation. We also defined a region of uncertainty around the

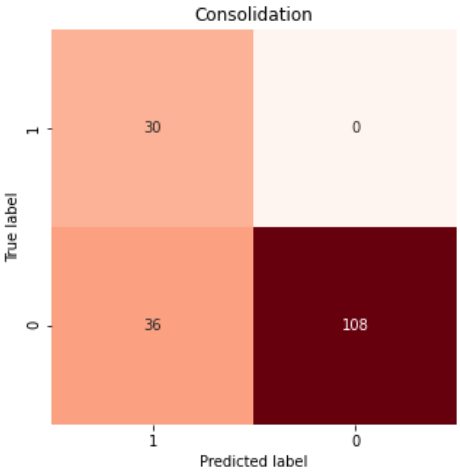
threshold, in which the model abstains from taking a decision, allowing it to express its opinion only when it's sure about it. In this way we try to limit as much as possible the number of incorrect predictions that, in this field, could be very dangerous. As we can see, the best performance is that involving Pleural Effusion, where the model is able to correctly classify 170 out of 202 diseases. The worst confusion matrix, instead, is probably the one involving Cardiomegaly, where there are 31 False Negative (i.e cases in which the disease in present but the model is not able to identify it).



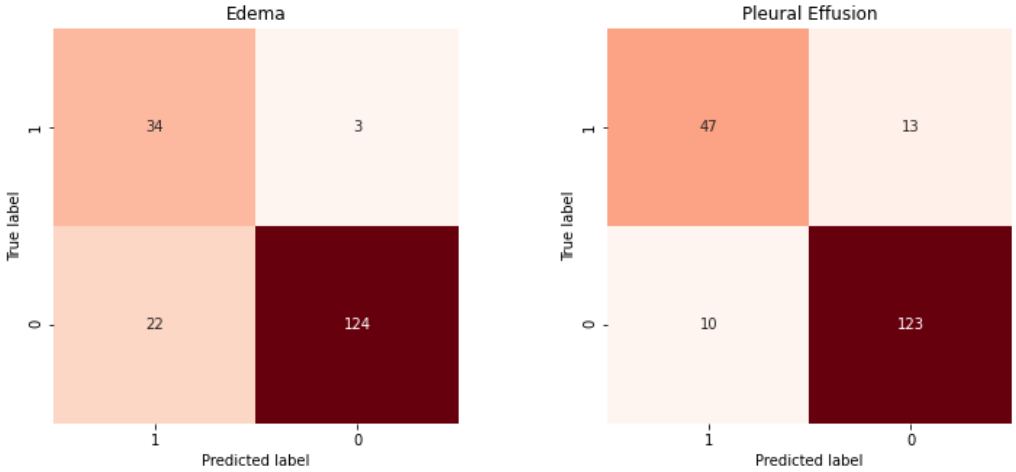
Uncertain: 25



Uncertain: 11



Uncertain: 28



Uncertain: 19 Uncertain: 9

Figure 5.14: Confusion Matrices

5.4 Class Activation Map

We will now show some samples of the Class Activation Maps that have been generated. Differently from previous works, we also provided, together with the CAM, a method to automatically extract a bounding box surrounding the area interested by the disease. Given the fact that it's hard to evaluate their goodness without having a medical background, we checked out how the model behaves in localizing the Support Devices, more easily identifiable also by a non expert eye. In Figure 5.15 we can clearly see that the model is able to surround the medical device positioned in patient's chest.

CXR CAM Bounding Box

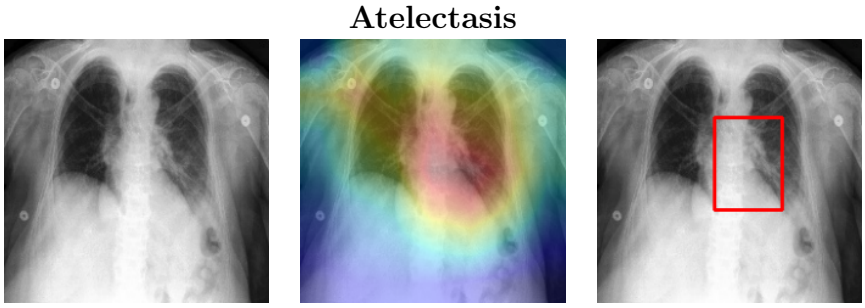
Figure 5.15

CXR CAM Bounding Box

Figure 5.16

CXR CAM Bounding Box

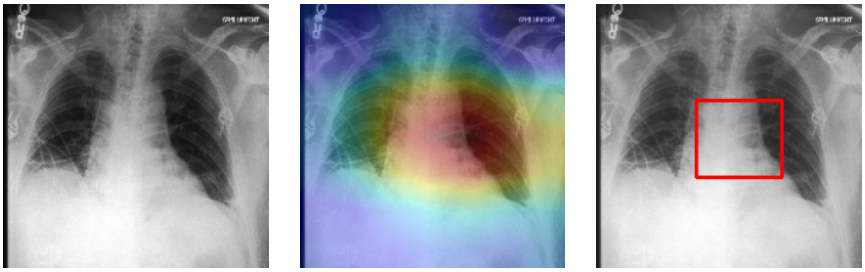
Figure 5.17



CXR CAM Bounding Box

Figure 5.18

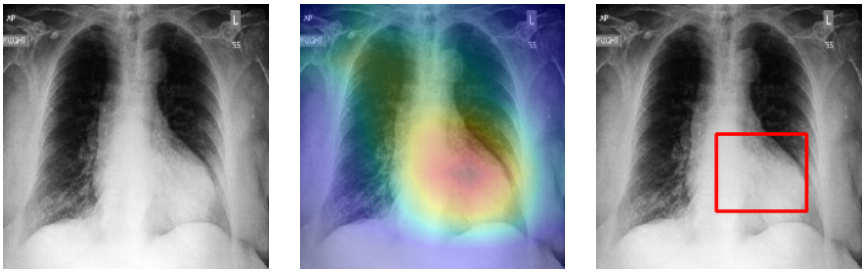
Atelectasis



CXR CAM Bounding Box

Figure 5.19

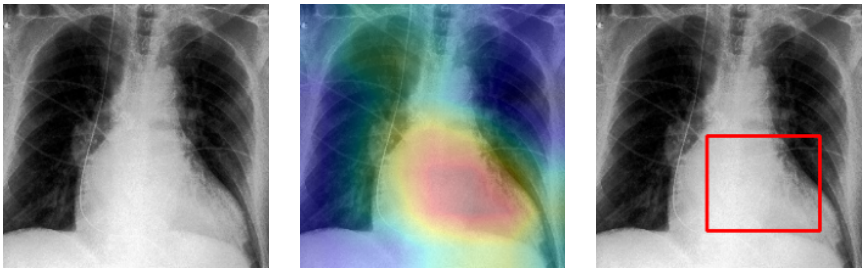
Cardiomegaly



CXR CAM Bounding Box

Figure 5.20

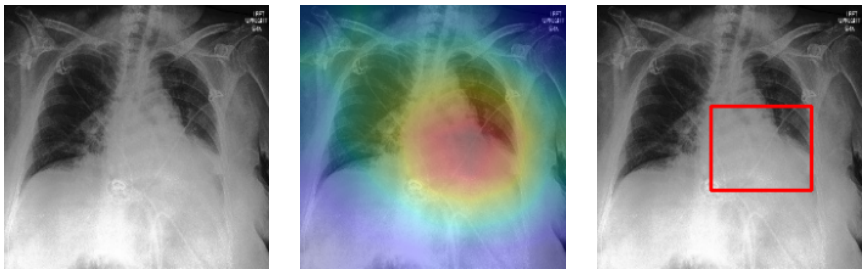
Cardiomegaly



CXR CAM Bounding Box

Figure 5.21

Consolidation



CXR CAM Bounding Box

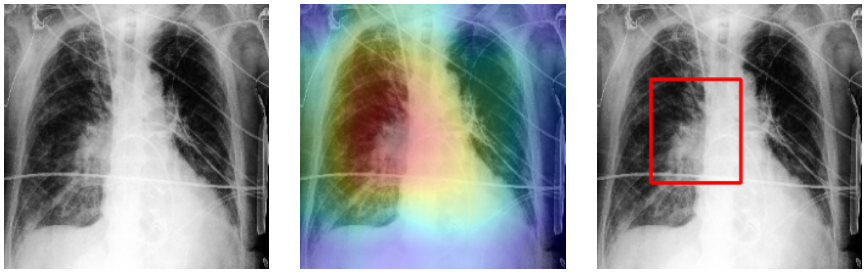
Figure 5.22

Consolidation

CXR CAM Bounding Box

Figure 5.23

Edema



CXR CAM Bounding Box

Figure 5.24

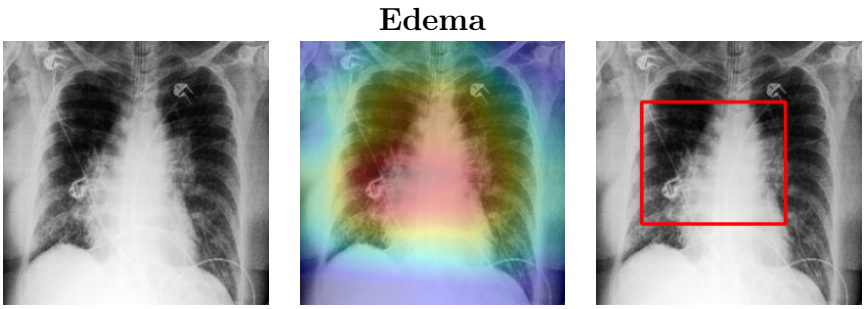


Figure 5.25

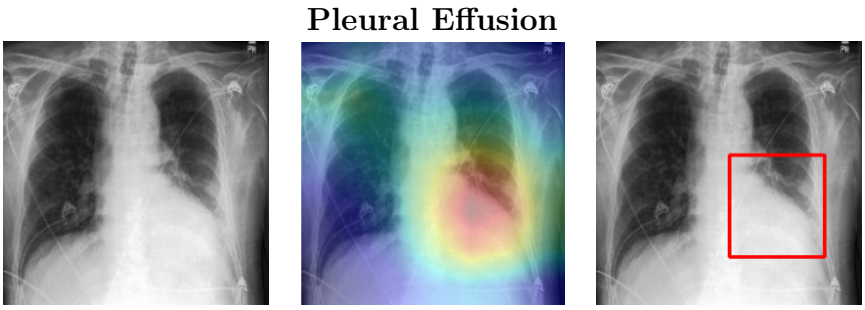


Figure 5.26

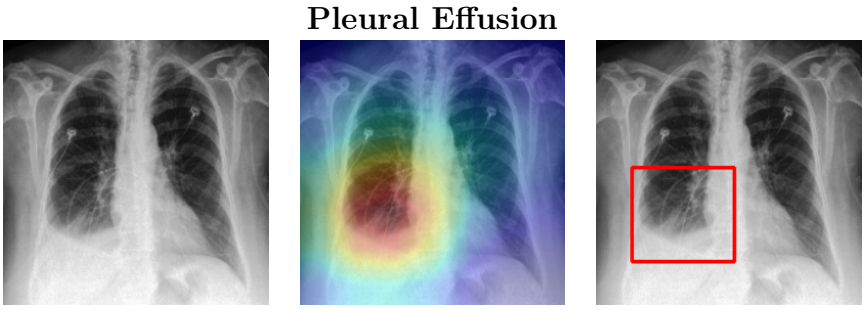


Figure 5.27

5.5 Summary

At the beginning of this chapter we have presented the evaluation criteria we have used to assess the performance of our models. Then we have presented all the results we achieved using the approaches defined in the previous chapters, discussing them

and comparing each other. Using the confusion matrices we tried to give the lecturer a more comprehensible summary about our model performances. We then compared our results with that obtained by the state of the art. Finally, we have provided some samples of the CAMs and Bounding Boxes that have been generated. In the next chapter we will give some considerations about our overall work and we will discuss some possible future works related to it.

Chapter 6

Conclusions And Future Works

In this chapter we will conclude our work, giving some general considerations about what we have done and some possible future works.

6.1 Conclusions

In this work we have presented the problem of analyzing Chest X-Rays to automatically classify some pathologies. Starting from some works that were previously carried out by the scientific community, we developed a system able to achieve a mean AUROC across five different diseases of 0.902. Although Pham et al. [25] in their work obtained a better result, we introduced some novelties.

Starting from the techniques proposed by the related works mentioned above, we developed multiple models using U-Ones and Label Smooth Regularization to deal with uncertain labels and Conditional Training to exploit diseases hierarchy. Then we investigated different approaches to aggregate the different models, showing that, instead of making a simple average of the predictions, the use of a weighted average, based on models' entropy, can improve the ensemble performance. Then, using the embeddings extracted by means of the CNNs, we showed that is possible, taking less time and computational resources, to build a system able to reach (or even surpass) the performance obtained by neural networks. Then, using the confusion matrices we tried, differently from other works, to give a more comprehensible vision of our model's output, showing exactly the number of correct and incorrect predictions, allowing also the system to abstain in case it is very uncertain about its diagnosis. Finally, for what it concern the pathologies visualization, we tried to improve it by aggregating the different Class Activation Maps produced by the models and we proposed a system to automatically extract a bounding box surrounding the region that most probably is

interested by the disease.

6.2 Future Works

We will now talk about some approaches that could be investigated to improve this work.

In our research we tried to build a model that was as general as possible. However Pham et al. [25], in their work, used some performance tweaking that allow their model to achieve very high results, with the downside of losing some generalization capability. It could be interesting to check whether those performance tweaking would actually lead to a better result or not.

For what it concern data, as we already said in Chapter 3, during the preprocessing we discarded some informations related to the patients, such as its age and gender. These data instead, could be leveraged, trying to check if, giving them as additional input to a model, is possible to enhance the predictions. Moreover, in our work we only employed the frontal CXR, but the dataset we used comes with a pretty large number of Lateral CXR. A system able to process both a Frontal and a Lateral view would probably be able to better discriminate among the different diseases.

Moreover, as we have seen, one of the biggest issues related to Chexpert dataset is that it's quite unbalanced, causing the perfomance of the less represented class to be worse than the other. Using some class balancing algorithm could produce an overall improvement. For what it concern data, finally, it is worth noting that it exist a dataset whose CXRs have been labelled using the same approach employed in Chexpert [13], containing almost 370 thousand additional images that could be used to train the models (with the downside that this approach requires more time and computational resources)

As we already told, the embeddings are able to capture the meaningful information needed to identify the presence or absence of a disease, with the advantage of drastically reducing the input dimension. In our work we used them to train Random Forest and XGBoost. However, given the small dimension of the embeddings (compared to that of the original images), it would be possible to investigate any other Machine Learning algorithm, even those that initially seemed to be not suitable for solving the task of CXR image analysis.

Once the embeddings have been extracted, moreover, they could be used to train models able to find any other pathology, not only those present in CheXpert. As proof of concept, we used them to identify the Chest X-Rays affected by SARS-CoV-2, the virus causing a pandemic that, starting from December 2019, spread all over the world. Using our trained CNNs, we extracted the embeddings of almost one hundred CXRs coming from sick patients and we used them to train a binary classifier using XGBoost, with the aim of distinguishing a regular Pneumonia (whose corresponding CXRs were already provided by CheXpert) from one caused by SARS-CoV-2. We used 10-fold cross validation as evaluation approach, obtaining the results in Table 6.1.

Fold	**Accuracy**	**AUROC**
Fold-1	1.000	1.000
Fold-2	0.947	0.929
Fold-3	0.947	0.955
Fold-4	1.000	1.000
Fold-5	0.947	0.944
Fold-6	0.947	0.944
Fold-7	0.895	0.889
Fold-8	1.000	1.000
Fold-9	0.947	0.944
Fold-10	1.000	1.000
AVG	0.963	0.961

Table 6.1: COVID-19 Accuracy and AUROC of the 10 folds.

As we can see, we obtained an average AUROC of 0.961 without using the full images, but exploiting only the embeddings. In some cases the model is able to perfectly distinguish the two diseases, as we can see in the confusion matrix reported 6.1.

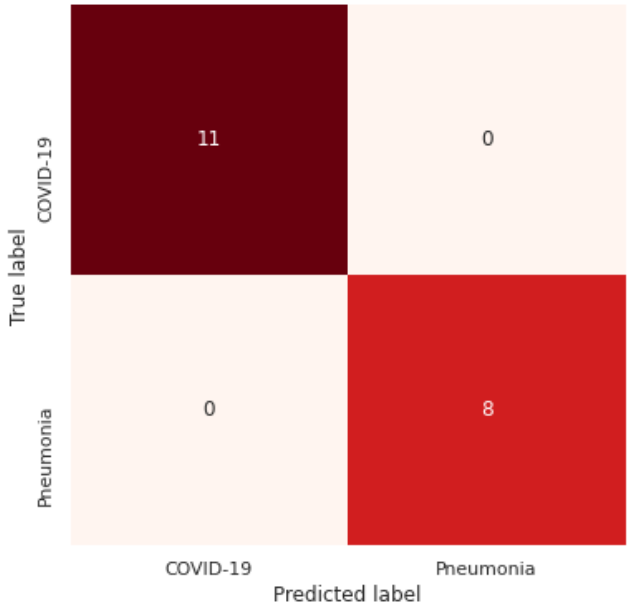


Figure 6.1: COVID-19 Confusion Matrix

Finally, the last approach we want to propose is that of using the bounding box that we generated to extract the area interested by the pathology and use it as input to the model, in order to let it focus its attention on a smaller region of the image.

Acronyms

AI	Artificial Intelligence
AUROC	Area Under Receiving Operating Characteristic
ANN	Artificial Neural Network
CAM	Class Activation Map
CNN	Convolutional Neural Network
CXR	Chest X-Ray
DL	Deep Learning
FC	Fully Connected
FCNN	Fully Connected Neural Network
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GAP	Global Average Pooling
LSR	Label Smoothing Regularization
ML	Machine Learning
NLP	Natural Language Processing
RF	Random Forest
TN	True Negative
TP	True Positive
TPR	True Positive Rate

Bibliography

[1] Klaus-Peter Adlassnig. "A Fuzzy Logical Model of Computer-Assisted Medical Diagnosis". In: *Methods of information in medicine* 19 (Aug. 1980), pp. 141–8. DOI: 10.1055/s-0038-1635271.

[2] Hosein Alizadeh, Muhammad Yousefnezhad, and Behrouz Minaei Bidgoli. "Wisdom of crowds cluster ensemble". In: *Intelligent Data Analysis* 19.3 (2015), pp. 485–503.

[3] Leo Breiman. "Bagging predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140. DOI: 10.1007/BF00058655. URL: https://doi.org/10.1007/BF00058655.

[4] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* (2016). DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[5] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2016. arXiv: 1610.02357 [cs.CV].

[6] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[7] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine." In: *Ann. Statist.* 29.5 (Oct. 2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451. URL: https://doi.org/10.1214/aos/1013203451.

[8] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st. O'Reilly Media, Inc., 2017. ISBN: 1491962291.

[9] Stanford ML Group. *CheXpert Competition*. https://stanfordmlgroup.github.io/competitions/chexpert/.

[10] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.

[11] Gao Huang et al. *Densely Connected Convolutional Networks*. 2016. arXiv: 1608.06993 [cs.CV].

[12] Jeremy Irvin et al. *CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison*. 2019. arXiv: 1901.07031 [cs.CV].

[13] Alistair E. W. Johnson et al. *MIMIC-CXR-JPG, a large publicly available database of labeled chest radiographs*. 2019. arXiv: 1901.07042 [cs.CV].

[14] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[15] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541.

[16] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. 2013. arXiv: 1312.4400 [cs.NE].

[17] Philip S. Maclin et al. "Using neural networks to diagnose cancer". In: *Journal of Medical Systems* 15.1 (1991), pp. 11–19. DOI: 10.1007/BF00993877. URL: https://doi.org/10.1007/BF00993877.

[18] Jayawant N. Mandrekar. "Receiver Operating Characteristic Curve in Diagnostic Test Assessment". In: *Journal of Thoracic Oncology* 5.9 (2010), pp. 1315–1316. ISSN: 1556-0864. DOI: https://doi.org/10.1097/JTO.0b013e3181ec173d. URL: http://www.sciencedirect.com/science/article/pii/S1556086415306043.

[19] John McCarthy. *What is artificial intelligence?* 1998. URL: http://jmc.stanford.edu/articles/whatisai/whatisai.pdf.

[20] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[21] Stephen Milborrow. *CART tree titanic survivors*. 2011. URL: https://commons.wikimedia.org/wiki/File:CART_tree_titanic_survivors.png.

[22] Selvaraj Naicker et al. "Shortage of healthcare workers in sub-Saharan Africa: A nephrological perspective". In: *Clinical nephrology* 74 Suppl 1 (Jan. 2011), S129–33. DOI: 10.5414/CNP74S129.

[23] Judea Pearl. “Bayesian Networks: a Model of Self-Activated Memory for Evidential Reasoning”. In: *Proceedings of the 7th Conference of the Cognitive Science Society, 1985*. 1985, pp. 329–334.

[24] Erkki Pesonen et al. “Diagnosis of Acute Appendicitis in Two Databases. Evaluation of Different Neighborhoods with an LVQ Neural Network”. In: *Methods of information in medicine* 37 (Jan. 1998), pp. 59–63. DOI: 10.1055/s-0038-1634497.

[25] Hieu H. Pham et al. *Interpreting chest X-rays via CNNs that exploit disease dependencies and uncertainty labels*. 2019. arXiv: 1911.06475 [eess.IV].

[26] Pranav Rajpurkar et al. *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning*. 2017. arXiv: 1711.05225 [cs.CV].

[27] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[28] Edward Shortliffe. *Computer-based medical consultations: MYCIN*. Vol. 2. Elsevier, 2012.

[29] A Shustanov and P Yakimov. “Modification of single-purpose CNN for creating multi-purpose CNN”. In: *Journal of Physics: Conference Series* 1368 (Nov. 2019), p. 052036. DOI: 10.1088/1742-6596/1368/5/052036.

[30] Basilio Sierra and Pedro Larrañaga. “Predicting the Survival in Malignant Skin Melanoma Using Bayesian Networks Automatically Induced by Genetic Algorithms - An Empirical Comparision Between Different Approaches”. In: *Artificial Intelligence in Medicine* 14 (1998), pp. 14–215.

[31] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[32] Friedrich Steimann. “On the use and usefulness of fuzzy sets in medical AI”. In: *Artificial Intelligence in Medicine* 21 (2001), 131ą137.

[33] Christian Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: 1602.07261 [cs.CV].

[34] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV].

[35] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. Aug. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[36] Steven Walczak and William J. Nowack. "An Artificial Neural Network Approach to Diagnosing Epilepsy Using Lateralized Bursts of Theta EEGs". In: *Journal of Medical Systems* 25.1 (2001), pp. 9–20. DOI: 10.1023/A:1005680114755. URL: https://doi.org/10.1023/A:1005680114755.

[37] Xiaosong Wang et al. "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)* (July 2017). DOI: 10.1109/cvpr.2017.369. URL: http://dx.doi.org/10.1109/CVPR.2017.369.

[38] L.A. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(65)90241-X. URL: http://www.sciencedirect.com/science/article/pii/S001999586590241X.

[39] B. Zhou et al. "Learning Deep Features for Discriminative Localization." In: *CVPR* (2016).