UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

DEEP LEARNING STUDY FOR PULMONARY ILLNESS DIAGNOSIS

HUGO ARANDA SANCHEZ

Thesis supervisor
AGUSTÍN FERNÁNDEZ JIMÉNEZ (Department of Computer Architecture)

Degree
Bachelor's Degree in Informatics Engineering (Computing)

Bachelor's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

21/10/2024

Abstract

English:

This Bachelor's Thesis will explore different Deep Learning categorization possibilities in order to solve the pulmonary illness diagnosis problem using chest RX scans. Creating an implementation of a state of the art model in with GPU acceleration in a HPC-like environment. In order to do so, extensive background on the state of the art techniques covered as a compendium of sorts to synthesize years of knowledge and breakthroughs. The proposed solution will be presented in well packaged code with plenty of customization options and flags for swift hyperparameter testing. The selected computer architecture used is a single GPU machine with a *RTX 3060* with Tensor Core Architecture and 12 GB of VRAM used as an accelerator. Finally, the thesis contains Classification performance and Efficiency studies to test and explore the alternatives thoroughly. Thus, the result of the thesis will be a series of informed design decisions and knowledge on models related to this problem.

Español:

Este trabajo de final de grado explorará las diferentes soluciones de Deep Learning aplicadas al problema de diagnóstico de afecciones pulmonares utilizando radiografías de pecho. Por tanto, se creará una implementación de un modelo comparable al *state of the art*. La solución estará dotada de aceleración con GPU mediante un entorno extrapolable a HPC. Con tal de lograr estos objetivos, un extensivo estudio de las técnicas actuales será realizado. Resultando en una síntesis del conocimiento sobre el problema y posibles soluciones a modo de marco teórico. La solución se presentará en un código apropiadamente formateado con opciones de configuración mediante *flags* que permitan modificar de forma sencilla la arquitectura. La máquina seleccionada para la tarea es un ordenador con una única gráfica *RTX 3060* con arquitectura dotada de Tensor Cores y 12 GB de VRAM que será utilizada como acelerador. Finalmente, el trabajo contendrá una experimentación completa centrada tanto en la capacidad de clasificación, como la eficiencia de las soluciones para comparar de forma informada las posibles configuraciones. Por ende, el resultado del trabajo consiste en una serie de decisiones de diseño informadas y conocimiento sobre los modelos.

Català:

Aquest treball de final de grau es dedicarà a explorar les diferents solucions d' Aprenentatge profund per aplicades al problema de diagnòstic d'afeccions pulmonars utilitzant radiografies de pit. Per tant, es crearà una implementació d'un model comparable al *state of the art*. La solució serà compatible amb acceleració amb GPU mitjançant un entorn extrapolable a HPC. Per aconseguir aquests objectius, es realitzarà un extensiu estudi de les tècniques actuals. Produint una síntesi de coneixement amb relació al problema i les possibles solucions a manera de marc teòric. La solució presentarà un codi apropiadament formatat amb opcions de configuració mitjançant *flags* que permetran modificar de manera senzilla l'arquitectura. La màquina seleccionada per a la tasca és un ordinador amb una única gràfica *RTX3060* amb arquitectura dotada de Tensor Cores i 12 GB de VRAM que s'utilitzarà com accelerador. Finalment, el treball es contindrà una experimentació completa centrada en capacitat de classificació i eficiència de les solucions per comparar de manera informada les possibles configuracions. Per tant, el resultat del treball consisteix en una sèrie de decisions de disseny informades i coneixement dels models.

Key Words: Deep Learning, Chest X Ray, Imbalanced Dataset

Contents

1 Introduction and Context

Humans have an inherent ease to perceive heaps of information from their surroundings effortlessly. In one glance we can grasp complex data around us like distance and size of objects through perspective or more context related aspects like pattern recognition. It is perhaps because of this natural talent we have to understand three-dimensional spaces and classify elements within them that historically this problem was thought to be easier in the realms of Artificial Intelligence[1].

This can't be further from the truth, bearing in mind that to this day Computer Vision is one of the most difficult problems of Computer Science. Basically, our main objective in Computer Vision is to emulate that effortless capability of describing the world trough the different properties that it's composed of. Properties such as shape, depth, pattern recognition and texture. Hence, we try to emulate artificially and automatically that human process of data extraction from a single image or a selection of them.

We can find a lot of examples of this type of AI being used nowadays such as *Character Recognition*, *Self driving vehicles*, *Biometrics* and *Motion Capture* among others.

On a more positive note, lately this area of study has seen a swift progress thanks to the success of deep learning bringing a lot of performance and utility to computer vision. Unfortunately, deep learning models are very resource hungry in their training since they revolve around the processing of big datasets to learn from them. That fact together with the rise of the presence of High-Performance Computing (form now on HPC) has lead to a very prolific relation between the two fields making it possible to speed up the process of training the model [2].

This Bachelor's Thesis aim to produce a implementation that exploits the 2017 introduced Tensor Cores in GPUs that allow the parallel architecture of GPUs to be used for deep learning purposes[3]. Opening a new frontier in general computing and deep learning in particular that makes a big step closer to better performances in Computer Vision.

Finally and referring to the actual model selected and implemented we have to look for the medical field. Computer Vision has intersected naturally with medicine, providing new solutions to long-lasting problems. Bringing the capabilities of being able to extract automatically geometric and biological properties of different tissues trough image processing, helping with diagnosis among other aspects[4].

One of the most readily available images in the medical field is the chest X-ray since it's a fairly common and simple test that is carried out regularly and brings a lot of information about many lung and heart diseases. This brings the perfect breeding ground for a great dataset on which deep learning can show promise. It is for this specific reasoning that the selected dataset for the project will be ChestX-ray8 comprised of more than a hundred thousand chest RX images associated with different pathologies[5].

1.1 Academic Framework

Within the Bachelor Degree in Informatics Engineering taught at FIB (Barcelona School of Informatics), this project theme resonates with the specialization of Computer Science. More particularly related with the graphics, computer vision and High Performance Computing area.

The range of transversal key competences covered by the project will be aligned with those expected of a Computer Science specialized Informatics Bachelor. As such the main objective is to provide with an efficient solution bearing in mind the current state of the art.

Mainly, aspects related to the application of fundamental principles and computation models. Such as evaluation of computational complexity in order to implement the best performance solution and the evaluation and the selection of platforms to develop and produce software solutions. Since a state of the art evaluation and adaptation of current models will be carried out in order to implement the final solution.

Also there will be aspects related to the effective and efficient development of algorithms for complex computational problems (Diagnosis trough computer vision). Specifically, knowledge on intelligent systems paradigms to build a computer system related to the field. With special stress on the computational learning with large data volumes.

Finally, concepts related to the developing of computer solutions that take into account the computer architecture where they are executed. Implementing for example critical code evaluating different criteria and taking into account the underlying hardware architecture on which the program runs on[6].

1.2 Relevant Concepts and Terms

Now we will briefly introduce the relevant concepts, techniques and systems related to the project that will help introduce the background presented later in the memory.

Neural Networks

The building block of the neural network is the neuron, understood as a vague abstraction inspired in real neurons, but extremely simplified. The neuron is a computational unit that sums several numerical inputs modified by a weight multiplier. The values are added and a special value called *bias* is added after. Then it has a *activation function* that decides bearing in mind the calculated value if the specific neuron *activates* or not and outputs another floating-point value. A visual representation can be seen in figure 1.

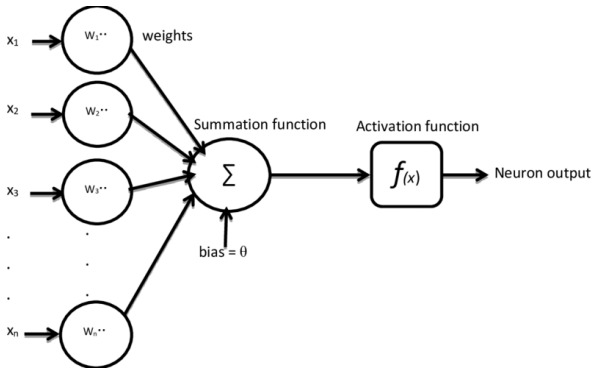


Figure 1: Visual representation of a neuron extracted from [7].

Then with these individual neurons we can arrange networks like with real networks connecting the different outputs and inputs accordingly. That is on a very basic note the definition of a *neural network*.

In machine learning the main structural feature of neural networks is that specific groups of neurons are arranged in layers so that they get inputs from the previous layer and put some work into them that transform the information given by the latter ones creating a flow of information (*feed-forward nework*). This together with clever distribution of several layers with several neurons creates the artificial intelligence behaviour we expect. The learning process involves the gradual improvement of weights trough iteration [8].

Tensor

A tensor is a collection of numbers in block form with a number of dimensions and size for each dimension. This is a broad definition that's used in Machine Learning to describe any sort of block of data coming in or out of a particular layer. Since they can have any sort of dimensionality they can describe simple 1D elements like lists, 2D elements like grids or even 3D volumes depending on what the layer works with.

Convolutional Neural Networks

Convolutional Neural Networks are a specific type of neural network which are particularly useful when working with grid-like topology such as images. In conventional multi-layer neural networks the input is comprised of a vector, that doesn't pair very well with higher dimensional data inputs. Since we might lose in vectorization context of the data in neighboring elements in those higher dimensions. That's why in convolutional neural networks an extra convolution step is added in order to preserve that type of information throughout the different layers as seen in figure 2.

Convolution as understood for Neural Networks deviates a bit from the pure mathematical concept. Intuitively, and for the purpose of general understanding, convolution resembles somewhat to applying a *filter kernel* that sweeps over the input producing an output (or feature map). This filter can be layer of neurons with particular weights and activation functions that perform a set operation to each pixel in parallel. A neuron on a filter can process a set of pixels gathering information from a certain neighbourhood detecting local features.

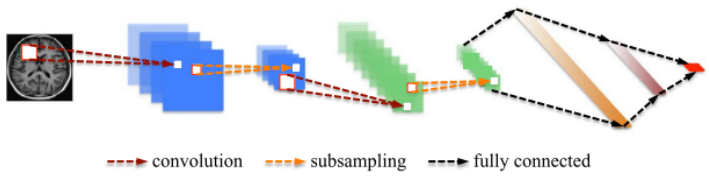


Figure 2: Example of the structure of a CNN extracted from [4].

When a tensor moves trough convolution layers it might change it's size and dimensions.

Tensor Core

Tensor Cores are a technology developed by NVIDIA. Introduced in 2017 under the Volta GPU architecture. They have been a staple of GPU for general processing bringing a new instruction of matrix-multiply-and-accumulate on 4x4 matrices on a single clock cycle. This particular instruction is very relevant in computing the previously described behaviour of a neuron treating the tensors that the neural network operates with [3].

1.3 Problem Description

This Bachelor's Thesis will develop a particular implementation for a GPU capable architecture that will be able to classify thoracic X-ray scans among eight different possible affections. The aim is to produce an efficient model that achieves state of the art accuracy efficiently and creates an aid system for the diagnosis process learning from the training data. The different affections that are contemplated in the dataset are all lung or heart related. Some are not diseases by themselves but an indicator of other more complex problems that have to be looked into (like Cardiomegaly). The classifying task will take into account this fourteen common thoracic diseases: Atelectasis, Cardiomegaly, Consolidation, Edema, Effusion, Emphyisema, Fibrosis, Hernia, Infiltration, Mass, Nodule, Pleural Thickening, Pneumonia and Pneumothorax.

This project will explore the current state of the art in order to select the best solution for our particular machine and try to explore experimentally how they behave. Convolutional neural networks have been at the spotlight on AI investigation for some time so there are plenty of different solutions to scour in order to find the best fit.

1.4 Agents Involved

The resulting system can be of value to *other researchers* that are involved with the use of neural networks since experimental findings can be of great help to start a project with previously collected knowledge.

More specifically, this project adds particular value to *doctors* that work with *RX diagnosis*. Specially after the pandemic of 2020 there's a special stress to the study of lung affections and the long term physical damages that the population might have. The study and implementation of efficient models to manage autonomous diagnosis can be of great help when trying to do mass testing to the population to prevent sanitary issues. This can be of help to stakeholders that need to process a lot of medical data with small groups of professionals to aid the process. This can be of great interest to *directors of hospitals* in order to help their workers detect and treat more efficiently different illnesses.

Finally, it also has a particular interest for the *Computer Architecture department at FIB*. Since the project will be carried out in boada or a similar HPC GPU based architecture. In the end development on the Boada engine was discarded but the architecture of the selected GPU is identical to the Boada one so portability should be easy in future projects.

2 Justification

In terms of justifying the project, the dataset selected has a lot of papers that refer to it creating diverse types of solutions so we will see how they hold in order to focus our study in something that's worth studying. The paper presenting the ChestX-ray8 dataset[5] was presented on the Computer Vision and Pattern Recognition Conference in 2017, which is a very influential conference on the field sponsored by the IEEE Computer Society and the Computer Vision Foundation[9].

2.1 State of the art sample

Many different approaches using different types of neural networks can be found:

- CheXNet[10]: is a 121-layer convolutional neural network trained on the ChestX-ray14 dataset (an extension of the ChestX-ray8 with enhanced classes that will be seen in other models). This model exceeded the average radiologist performance on F1 metric at classification tasks. It's one of the first models to use the dataset and already achieved state of the art results for the field detecting Pneumonia.

- CheXClusion[11]: is a 121-layer densely connected convolutional network similar to the first one with a particular stress on the study to try to find if protected data (Age or Gender) has some correlation with the inferred labels.

- ResNet50 (Global-Local)[12]: is a convolutional neural network with 50 layers of depth which can be specially pre-trained with the ImageNet database[13]. That way the learning task is a bit jump started. This approach was particularly interesting because the paper does not use a fixed set of predefined categories in order to pick anomalies outside the predefined set (thus getting a sub par AUC of 0.7833 but still impressive for the context given).

- NSGANetV1-A3[14]: is a particular case of convolutional neural networks where the network design process was automated reaching competitive performances at image classification with an evolutionary algorithm to search the neural network architectures. This process reaches an impressive AUROC of 0.847.

The particular network selection will depend on both ease of implementation and feasibility and original objectives unique to our approach. The selected backbone model will be similar to one of the previous ones with permutations to aid out study towards our objectives. The objective is to build from the state of the art onwards and test how new solutions can help this classification problem.

In this thesis case the main point of contention will be between ResNet50, DensNet121 and EfficientNet with some slight modifications. Since more sophisticated networks are out of the scope considering that the thesis author was inexperienced with the topic at the start of the project. Because of that, deep understanding of sleek and simple but powerful networks is considered of great value.

In order to gain that understanding, some experimentation and tinkering with the possibilities of the models will be done. The experimentation will take a deeper look upon the solutions that might help the model achieve better classification for less favourable classes due to class imbalance[15] with a multi-label setting (not conventional for a multi-class dominated field). Some general and straight forward hyperparameter tuning and data visualization will also be necessary and interesting to do.

2.2 Selection of tools

All of the previously stated models are created using the most common open-source platforms that provide a framework to prepare data, train models and implement them on GPU architectures. Those are *TensorFlow* and *PyTorch*. Both frameworks have to be considered for the project. Generally PyTorch is regarded as a more python friendly type of framework that offers a dynamic process of prototyping. While TensorFlow is better classified as a large scale and production oriented framework that excels in large deployments. At low level both TensorFlow and PyTorch use the *CUDA* framework to interact with the NVIDIA GPUs.

In the particular scenario of the papers observed we can see a small favor towards PyTorch. Nonetheless, both options will be considered on initial stages of research.

In terms of application behaviour monitoring the *Nvidia Nsight Systems* [16] performance analysis tool will be used to visualize the traces of the execution. That way we will be able to include computational analysis of the code in order to understand the theoretical differences of the different models and architectures with metrics and traces.

3 Project Scope

In this section the objectives of the project will be presented, discussed and evaluated. Given that the time has to be limited to accommodate to 18 ECTS an effort will be made to define accurate objectives that can be achieved.

General goal of the project

The main goal for the project will be to create a state of the art implementation of an deep learning architecture that configures different solutions into an optimized approach by iterative experimentation. Thus, the main objective will be producing that software solution taking into account different results of reproducible and ell documented experiments.

Sub-Objectives

The sub-objectives that comprise project then will be related to the implementation and parallelisation of the process:

- **To investigate the state of the art exhaustively** to search for the adequate model or set of models that will inspire the implementation.
  - To get a solid understanding of neural networks beyond the libraries

  - To learn about particular solutions on this field

- **To create a testing environment with GPUs as Accelerators** to ensure local testing environment is similar to HPC architectures using NVIDIA GPUs as Accelerators (correct libraries, virtual environments and drivers).
- **To investigate and develop an efficient training method** for the model in order to exploit correctly the GPU architecture and minimize useless trianing.
- **To investigate and develop the initial draft of the model** in order to get adequate state of the art scores on the classification metrics.
- **To develop initial draft into a competitive model** and try to find potential bottlenecks or upgrades within the proposed solution (iterative process).
- **To integrate explainable AI methods with the model** in order to try and dissect why the model decides what it does.
- **To create a model aware of the limitations of it's dataset** through previous visualization of the dataset and exploration of similar problems and possible solutions. The solutions might be developed for the problem or be outside of the medical imaging world. Thus using general CNN knowledge and investigations to aid the specific problem.

3.1 Requirements

The requirements of the software solution developed basically revolve around the scope of the project and resources given and the state of the art status that is targeted.

Functional

- Functional HPC environment with the ability to run, profile and test code that runs in GPUs parallel architecture.
- Secure environment against power loss and other type of disruptive events that can happen while training in order to not do futile executions.
- Performance in terms of classification quality have to be at least in the mix with the other projects alike bearing in mind that those projects might be the result of longer and more resource intensive efforts.
- Ease of switching between different model configurations through flags to create a diverse architecture that can take many forms without too much effort.
- Implement and source from other works a variety of different code solutions in terms of architecture. With at least three to four options for the main elements and hyperparameters within the configuration. With a particular interest on class imbalance and the process of learning.

Non-functional

- *Maintainability:* The model has to be able to sustain an iterative process and be upgraded without being a very expensive or resource intensive process to keep the thesis under budget.
- *Analyzable:* The product has to be able to be probed and studied by the trace to be able to develop the level efficiency evaluation intended.
- *Efficiency*: As a result of the analyzing process it is expected to gain efficiency or at least understand better the tough parts in order for future studies to benefit from the findings.
- *Transparency:* The considered solution will aim for better transparency within a very dark box type of programming that is deep learning with visual explanations of the models inner workings through CAM like algorithms [17].
- *Open Source:* As with all the knowledge and expansive amount of previous solutions considered for the project, the results of this thesis work will be open source in order for them to enlarge the collective knowledge on this matters.

3.2 Risks and Obstacles

All projects have to assume certain risks and try to come up with ways of being able to mitigate or avoid them whatsoever. Here there are a selection of the obstacles to overcome and possible risks that the project could face.

Obstacles

- *Inexperience:* For the author as an alumni it will pose a challenge to implement a neural network to such an extent since the previous experience on the field was very trivial and didn't require a deep understanding of the intricacies of Neural Networks.
- *Memory allocation*: The boada node is shared between different scholars, professors and students so allocating memory for the more than 40 GB that the dataset will pose an obstacle to overcome. In order for this circumstances to impact the project as little as possible we will try to start with petition processes as soon as possible.

Risks

- *Time Planning*: The author of the thesis has a 25h job on top of a full set of 30 ETCS to take for the spring semester (taking into account the time allocated for this Thesis) which can difficult the time dedication schedule making very time management a very critical point in the process.
- *Complexity of the problem:* As it has been discussed over the initial parts of the document Computer Vision is inherently difficult in the world of AI and there might not be a lot of room develop upon the already found solutions significantly.

- *Human error:* Since Deep Learning programs are resource intensive, unnoticed errors on running programs can lead to long, expensive and inconclusive executions. A mindful and exhaustive approach will be required in order to not waste many executions on such power hungry devices.
- *Health issues*: The author of the thesis is recovering from an intervention and is still not a 100% recovered. So pain and physical discomfort are a factor that affects the author's capabilities. Recovery should be on it's way but any further problems can compromise the development of the project.

4 Development

In terms of development the project will be comprised of two well differentiated phases required to produce the desired output. First of all, a necessary investigation and self-learning task will have to be done by the author bearing in mind the scope of the project requires a deeper knowledge on new topics. Afterwards, an implementation phase with iterative prototyping will take place with constant evaluation and efficiency analysis upon getting executable desirable results.

4.1 Methodology

4.1.1 Agile Methods

Since the author is not familiar with area of study a dynamic methodology that doesn't require extensive prior knowledge and allows for readjusting objectives according to context and results along the way is needed. In that context, it is needed to be able to redefine the process itself as it unfolds opting for the flexible workflows that agile methodologies bring.

If we take into consideration the time constraints and organization requirements a very adequate agile methodology would be Scrum. Since that methodology has an iterative and incremental nature to the delivery of products that goes well with the intended development path expected for the project. The project will be broken down into sprints (small periods of two weeks) that will have particular objectives. When reaching the end of any particular sprint the progress achieved will be measured and readjust the future plans to compensate underachieving sprints.

In that sense a complete sprint plan will be scheduled with the hopes of being executable by itself but with the right to review at any point of the process and readjust the scope of future sprints.

4.1.2 Practical Methodology Changes Justification

The project was initially thought to be developed from home with the help of the Boada client that allows for remote connection. In the end, development and testing were executed within a local PC of the thesis author with very similar environment features to the Boada system. This decision is due to a handful of key points that ended up shifting the objective

architecture from server to local PC with HPC configuration. The main reasoning is the following:

- Boada ran **consumer-grade gaming GPUs** similar to what you could find in a home PC. In that sense no extra performance or interest in testing that equipment was particularly appealing. Both architectures are from the Ampere family and have similar tensor core counts and potential FLOPS.
- Even tough Boada's *RTX 3080* is a couple tiers higher than the local *RTX 3060* GPU of the thesis author in the end the **local GPU had higher VRAM** that is critical in order to fit bigger models and batches to the training process which is a great advantage that leads to lower compute times.
- Multi-GPU can be useful for enhanced parallelization but it ultimately makes it harder for the user to pinpoint specific events within the execution of the trace. Therefore **multi-GPU hard to obtain traces** made the configuration not as favorable as initially thought.
- Profiling methods in server-like architectures are hard to use and have **limited expressive power**. The proposed *Nvidia Nsight* was vital for profiling and could not be installed properly on the cluster.

To sum up, the extra difficulties and hardships of working with the ssh accessed cluster didn't yield many improvements in terms of execution times, insights gained from the hardware or particular interest for the project (bearing in mind that the thesis author is a Computer Science major and not a Computing Architecture major).

As it was commented on the Agile Methods section, a strictly incremental design pattern will be carried out with intermediate functional versions that serve as benchmarks of their own for future developments. Plus, the incremental approach makes it so the rising complexity is highly manageable specially for an inexperienced developer. Since small changes can have big implications as a learning experience and a reproductibility test exercise produces great information to document in this memory. Because of that experiments will include all this incremental versions with deep sensitivity analysis through the parameters and configurations with remarks on computational performance.

As such, a significant part of the value of the thesis comes from reinforcement of the last discoveries with experimentation related to the performance of the different configurations. It also has some original remarks due to the experimentation found set different parameters and particular implementation proposed. Even though, when cited some of the implementation can have extracted code from breakthrough papers that it inspires from in order to be able to add as many technologies as varied as possible with already handcrafted code that fits the models and proposed architecture.

4.2 Used Resources

No special software licensing will be necessary since documentation will be executed with LaTeX. In the realms of programming and delivery all means will be open-source both for ease of implementation on a low resource project and extra control since open code allows to tweak minor effects to a greater extent.

The particular HPC initially proposed provided by the Computer Architecture department is the Gigabyte G492-H80 [18] arranged in a setup with 4 NVIDIA GTX 3080 and 3rd Gen. Intel® Xeon® Scalable Processors. The final PC on which the testing and development took place as it was described in the methodology was an *HP OMEN 40L GT21-0001ns*[19]. The PC had a RTX 3060 with 12 GB of VRAM and an AMD Ryzen 5 5600X.

In order to execute the desired methodology a monitoring tool used to manage the workflow will be *Trello*. The online application allows the creation of different objectives and to define and monitor the iterative process. Allowing for the two week sprint format to be executed and evaluated regularly.

Also since version control program will be needed in order to keep track of changes and be able to mantain the iterative process bearing in mind all changes done to the project so that if any problem is faced with a particular version we have earlier ones at demand. The preferred solution for this task by the author is *GitHub* since in the future the project will be open-source so that it can be used by the community to expand on the findings of this thesis.

4.3 Legality

In terms of the legality of the project, since we are working with a dataset that uses medical data it's important to determine the privacy of the images. This is specially important with image metadata offered. As stated on the original NIH website, the dataset was rigorously screened in order to remove all personal information between release. In the metadata analysis we see that no metadata related to the identity of the users besides their gender and age is stored.

In terms of licenses related to code usage all code inherited from other studies is strictly MIT Licensed to keep the project under the most unrestricted license possible to facilitate future studies to pick up what the thesis works on and iterate on it. The following MIT License Copyrights are necessary for the GitHub repository offered at `https://github.com/hugoarsa/TFG`: 2022 Naoki Katsura for Cosine Annealing, 2021 Alinstein Jose for Optimal Cutoff and GradCAM, 2020 Alibaba-MIIL for Asymmetric Loss [20].

5 Background

This section will introduce the key concepts and technical background necessary to understand the development of the thesis through the areas that are involved with it. All concepts and knowledge is from papers extracted from *The Computer Vision Fundation* and the literature referenced.

5.1 Deep Learning

Deep learning is a branch of machine learning and artificial intelligence focuses on learning from data with the use of **Neural Networks**. Particularly, it specializes in grasping complex features of raw data with a new representation that builds on those non trivial concepts through a combination of simpler elements. This frames the model within the **representation learning** realm, representing the knowledge extracted from the model with the neural network itself.

5.1.1 Artificial Neuron

The building block of the neural network is the **artificial neuron**, understood as a vague abstraction inspired in real neurons, but extremely simplified. The neuron is a computational unit that sums several numerical inputs modified by a weight multiplier. The values are added and a special value called *bias* is added after. Then it has a *activation function* that decides bearing in mind the calculated value if the specific neuron *activates* or not and outputs another floating-point value[8]. The mathematical relevance of having a non-linear function between neurons is mainly due to the *network collapse* problem. That concept relies on the fact that if no non-linear functions are added all the chain of small computations done by each neuron can be expressed in terms of the input and a chain of sums and multiplications that can be *collapsed* into a linear expression that can be computed on a signle neuron. Rendering all other neurons useless since they are not adding any valuable calculations that couldn't be done with just a single one. With non-linear activation information doesn't combine that way ensuring the expression capabilities of the network[8]. A visual representation can be seen in figure 3.

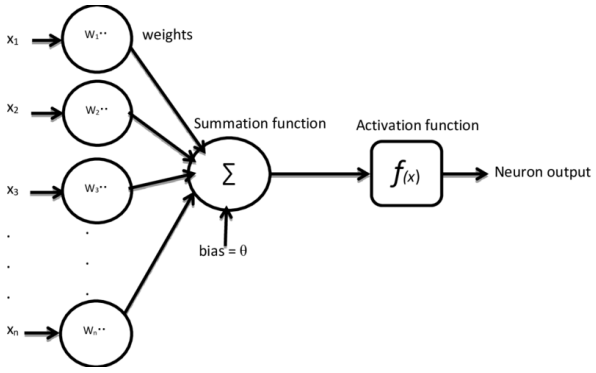


Figure 3: Visual representation of a neuron extracted from [7].

5.1.2 Artificial Neural Networks

Then with these individual neurons we can arrange networks like with real networks connecting the different outputs and inputs accordingly in order to produce one or more outputs that bear some type of significance. That is on a very basic note the definition of a *neural network*.

The main structural feature of neural networks is that specific groups of neurons are arranged in layers so that they get inputs from the previous layer and put some work into them that transform the information given by the latter ones creating a flow of information (*feed-forward network* or *Multi Layer Perceptron*). This together with clever distribution of several layers with several neurons creates the artificial intelligence behaviour we expect. The simplified structure of a neural network has: an *input layer* where data is fed, some amount of *hidden layers* arranged in a deliberate way to do relevant calculations, and an *output layer* that gives the final result.

5.1.3 Relevant historical Neural Networks papers

Looking back, the technology was first theorized with the McCulloch and Pitts paper at 1943 [21] with the first approach that tried to interpret natural neurons as a mathematical model. At first the technology was very simple and was thought to be fundamentally limited [22] since it wasn't able to solve non-linear separation due to it's single-layer neural network nature. This theory led to a particular disinterest in the technology until the discovering of a new learning procedure known as **back-propagation** that allowed for deep learning to occur. As stated in the original paper by Rumelhart, Hinton, and Williams "The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the met and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units."[23]. From that point onward, multi layer perceptrons (MLP) that captured non-linear features where able to be effectively trained and thus the Deep Learning revolution really took off.

5.1.4 Training Concepts

In order to train a Deep Neural Network or a general machine learning model some type of optimization has to be carried out. Since optimization revolves around minimizing some $f(x)$ function, we need a function that estimates the total error is necessary in order to asses how good the current model does in classification tasks. Such a function is called within the Machine Learning context as a *loss function*.

For multi-label codification tasks a typical loss function is the *categorical cross entropy* function as the distance between data and the model distribution represented by $f(x)_i$ and the training data as $\hat{y}$. The parameters of the model constitute the search space x and then minimizing the loss function results in an ideal set of parameters. In our case since neural networks use SoftMax [24] function to put the resulting model probability distribution between the range of $(0, 1)$ adding up to 1 total sum we can use a simplified version of the general function:

$$J(\theta) = -\sum_{i=1}^{n} \frac{e^{\hat{y}_i}}{\sum_{j=1}^{n} e^{\hat{y}_j}} \log\left(\frac{e^{f(x)_i}}{\sum_{j=1}^{n} e^{f(x)_j}}\right) \tag{1}$$

In order to minimize the function some minimal calculus concepts have to be put together to obtain the algorithm used, since the resulting loss function is highly non-linear and convex and linear solvers are not fisable. According to calculus the derivative of a function specifies the slope of $f(x)$ at a point x. In terms of training and machine learning a derivative can specify how to change x in order to make small improvements in y. The concept of a series of appropriately sized steps in x opposite to the derivative of the function to minimize $f(x)$ is called **gradient descent** and it's at the core of deep learning algorithms.

For functions with multiple inputs the concept of partial derivative $\frac{\partial f}{\partial x_i} f(x)$ can be utilized to refer as the derivative that measures how f changes according to changes to x_i on a specific point x. Finally the **gradient** is the derivative with respect to a vector that contains all partial derivatives for all variables at a point $\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)$. A final gradient related concept sometimes used is the **Jacobian matrix**. In order to define it we need a given function $f : \mathbb{R}^m \to \mathbb{R}^n$. Then, the Jacobian matrix $J \in \mathbb{R}^{n \times m}$ of f is defined such that $J_{i,j} = \frac{\partial f_i(x)}{\partial x_j}$[25].

In order to optimize different changes in parameters will affect the loss function we find with the gradients of the function with the model parameters and use some clever calculus to find the point of steepest descent moving to the negative gradient as in *gradient descent*. With that steepest descent we found a new point updating on the desired direction according to a **learning rate** η that produces the point $x' = x - \eta \nabla_x f(x)$ [25]. Eventually the objective is to converge in a local minimum of the given loss function.

Stochastic gradient descent sequentially updates parameters on the supposition that the gradient is an expectation that can be approximated with few samples. In this case minibatches are extracted from the training set with a particular set size. Since doing gradient descent with all data is highly computationally expensive SGD provides a safe way to keep the model learning at a consistent rate no matter the training size[25].

5.1.5 Modern Back-propagation

Backpropagation when being more precisely defined is basically a method for efficiently computing the gradients of the different points within the network in order to conduct the actual training algorithm that is stochastic gradient descent. Since numerically evaluating the gradient expression is highly expensive the backprop algorithm is vital for the process to be feasible to train networks.

Backprop can be generalized to the computation of any function's gradient but in our case our function will always be the loss function with the parameters of the network. Other derivatives needed might be necessary and can also be calculated through backprop.

In order to achieve this efficient calculus the chain rule of calculus has to be used to compute the derivatives by composition of other already known derivatives. Given $y = g(x)$ and $z = f(g(x)) = f(y)$ the chain rule is describes as the following

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} \tag{2}$$

Basically backprop computes the chain rule with an efficient order of operation that allows for swift finding of the desired gradients. In order to compute gradients the chain rule can be generalized for functions that map vectors and as such the resulting chain allows calculation of gradients with the previously obtained gradients. Assuming that $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, $g : \mathbb{R}^m \to \mathbb{R}^n$, *and* $f : \mathbb{R}^n \to \mathbb{R}$.

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^\top \nabla_y z, \tag{3}$$

where $\frac{\partial y}{\partial x}$ is the Jacobian matrix of g which is of size $n \times m$. Since we can obtain gradients x with Jacobian-gradient product any point in the graph can be deducted with this expression. Backprop algorithms have to take into account memory efficiency considerations related to the recalculation of already computed gradients throughout the recursive process of obtaining all gradients.

5.1.6 Overfitting and Underfitting

When training models the main interest is to train models that perform well with completely new and **generalized** examples that are not included in the training set. That means that the model is required to minimize the **training error** related to the *loss function* but also it needs to perform well related to the **test error** which includes new data that the model has never seen. According to this two errors and the concept of generalisation there are some events that can happen and need to be addressed in order for the models to perform at their maximum capacity.

The ideal scenario would be that the training error that we minimize is obviously small and that error translates accordingly to the test error having a small enough gap. If the second rule doesn't apply the event is called **overfitting** since the model learned only to recognize the training set but doesn't generalize accordingly. If the training error stays considerable there might be an **underfitting** problem related to an insufficient learning process with the training set resulting in the aforementioned error.

5.1.7 Multi label classification metrics

In order to quantify how good a classification model is there exists a need for certain classification metrics. **Precision** is the fraction of positive results that where correct and **recall** is the fraction of true positives on the test that where detected by the model. These two can be put in a curve that reflects the trade-off between them with the PR curve that can

be represented under a single number with the **F-score** calculated with the expression with precision as p and recall as r

$$F = \frac{2pr}{p+r} \tag{4}$$

5.2 Convolutional Neural Network

With Deep Learning now at the upfront of many technical challenges it has revolutionized the state of the art specially in computer vision realm with Convolutional Neural Networks (first uses attributed to Y. Le Cun [26]). Computer Vision problems benefit from the featureless feature extraction capability of neural networks.

Convolutional Neural Networks are a specific type of neural network which are particularly useful when working with grid-like topology such as images. In conventional multi-layer neural networks the input is comprised of a vector, that doesn't pair very well with higher dimensional data inputs. Since we might lose in vectorization context of the data in neighboring elements in those higher dimensions. That's why in convolutional neural networks an extra convolution step is added in order to preserve that type of information throughout the different layers4.

Convolution as understood for Neural Networks deviates a bit from the pure mathematical concept. Intuitively, and for the purpose of general understanding, convolution resembles somewhat to applying a *filter kernel* that sweeps over the input producing an output (or feature map). This filter can be layer of neurons with particular weights and activation functions that perform a set operation to each pixel in parallel. A neuron on a filter can process a set of pixels gathering information from a certain neighbourhood detecting local features.

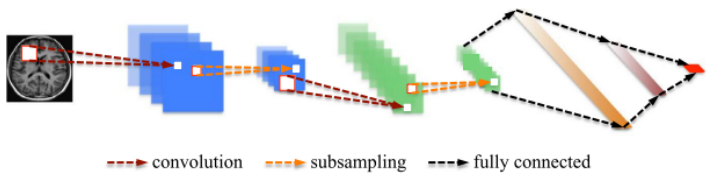


Figure 4: Example of the structure of a CNN extracted from [4].

Convolution layers as they are called detect local features in the feature maps with kernels that can be learnt through back-propagation. The operation is carried out taking all the feature maps (subsets of previous layer the size of the kernel obtained applying strides) and executing the convolution for each feature map. All this is put through the sum of a particular bias and non linear activation function as always.

After convolution pooling layers are needed to downsample the output feature maps of previous convolutions steps. Features in pooling are linked to the feature maps of convolution steps. Pooling layers find maximal values among the fields is convolved with (instead of doing matrix multiplication as in kernel multiplication). Stride is modified in order to not overlap the feature maps in this steps since we want to downsample unique data each time. This step allows for a reduction in size of features along the net in order to reduce computation steps[4]. Thereby, tensor size and shape change over the course of the forward pass.

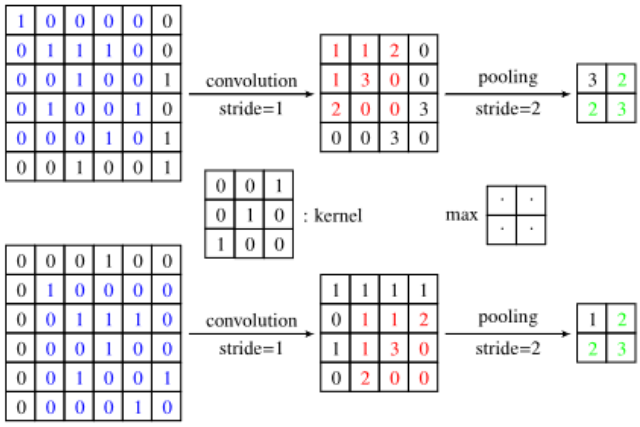


Figure 5: This example extracted from Deep Learning for Medical Image Analysis [4] (as well as the convolution explanation) shows a particular convolution and pooling execution. Also to be highlighted, the convolution-pooling step is invariant to translation since the green pixels remain the same despite the clear translation.

An example of CNN setting the state of art include the different ImageNet competition [27] breakthroughs like the 2012 winner AlexNet [28] with the importance of depth in layers when constructing models. Later the 2015 Szegedy et al. [29] pushed the state of the art with a rethinking of the structure of the neural network that revolved around using computation as efficiently as possible for big data scenarios. The paper added some ground rules to efficiently develop neural networks that are key to create better performing CNN at the cost of being more specific.

5.3 Existing models

Now that some ground concepts have been laid out for deep learning a deeper look on particular network architectures to see specifically how do they work and where there might be potential parallelization or optimizations.

5.3.1 ResNet

This subsection will be a comprehensible summary of the findings by He et al. 2015 [30] along with a breakdown of the ImageNet[13] detection contest winner model based on ResNet.

In order to appreciate relevance of this breakthrough it's important to understand that as networks got deeper there was a new limit that made them difficult to train once they reached a particular depth. This is highly interesting bearing in mind that more layers should result in a greater way of expressing possible permutations but experimental data extracted from the original paper seen in figure 6. This degradation is *not caused by overfitting* according to the experiments of the paper.

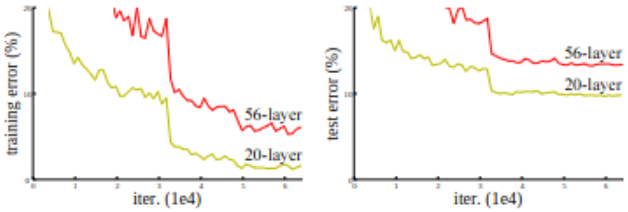


Figure 6: Results of experiments carried out by He et al. on the 2016 ResNet paper [30] that highlight the results of higher training error as the networks gets too big.

To put it more into perspective, an example is given where a shallower architecture with a particular model and a constructed deeper model that is just the previous shallow architecture with the *identity function* on the added layers. This expanded theoretical model is at least as good as it's previous iteration by default contrary to what the experimental data suggests.

This particular example inspires a new framework that revolves around the network being able to have that identity function as part of the information pipeline of the network as seen in figure 7. In order to achieve this, for a particular set of stacked nonlinear layers with an input x that produce a particular mapping of $F(x)$ an addition is performed between the mapping and the identity of the input. That function is called $H(x) := F(x) + x$.

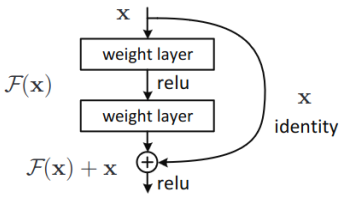


Figure 7: A visual representation of the resent building block extracted from the original He et al. 2015 paper [30] that showcases the identity function alternative path of the data flow.

Once defined $H(x)$ a **residual approach** is suggested so that the multiple nonlinear layers originate residual functions like $F(x) := H(x) - x$. Basically layers just train what difference there is between the original input and the next. This particular approach is due to the theorized initial example that suggested models have a hard time figuring out the identity operation. Since now an all zero weigh initialization has by default the identity.

In order to implement the **identity mapping** proposed at the building block a *shortcut connection* and element wise addition is used. This shortcuts allows for the identity function to go though a particular set of layers without adding any complexity. An important remark is that the dimensions of the input and output have to be the same in order to the addition to be performed (sometimes small operations are needed for the match).

The resulting architecture produces satisfactory results in a variety of datasets and contest granting first place to the model in many of them. The satisfactory results can be seen on the papers experiments in figure 8

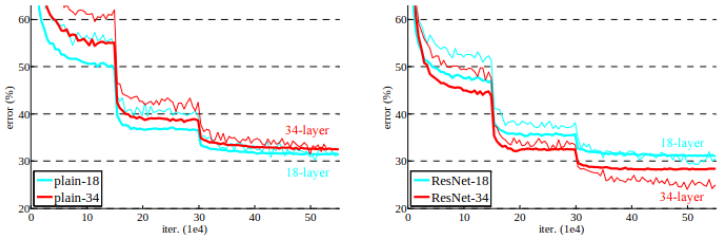


Figure 8: This is the final results of the proposed architecture of the ResNet extracted from the He et al. 2015 [30] that demonstrates not only no degradation as layers are added but also better results in general.

5.3.2 DensNet

Another interesting model in terms of efficiency is the DensNet model introduced by Huang et al. 2016 [31]. The proposed model reflected on the ground work proposed by the ResNet and others and found the common factor of shortening paths from early layers to later ones between all the previous solutions. With this concept in mind a new architecture is proposed that has all layers that have matching feature-map sizes connected though each other improving information flow. The resulting networks are more efficient since they require fewer parameters because feature-maps can be reused so that useful found functions in early layers can be used throughout the forward pass without being recalculated.

In order to understand how all this is done we can look at the example construction given at the original paper [31]. First an image x_0 is passed through the DensNet. The network has L layers that implement a non-linear transformation $H_l(\cdot)$.

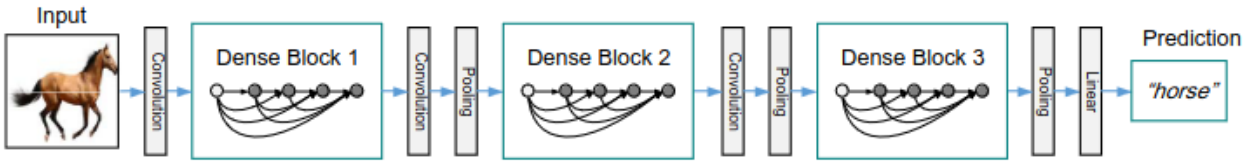


Figure 9: This is a visual representation of a DensNet comprised of 3 dense blocks that are densely connected to eachother. The image is extracted from the original Huang et al. paper [31].

In order for the improvement of information flow through the layers any layer connects to it's subsequent networks, creating a **dense connection**. Thus, each networks receives all output feature maps of preceding layers concatenated into a single tensor. The particular function $H_l(\cdot)$ is defined as in another He et al. 2016 paper [32] that improved efficiency in ResNets and also now in DensNets. To take into account the evolution of size that happens within the flow of the data through the convolutional network the dense sections occur between the pooling layers that downsize the feature-maps. These densely connected parts are called *dense blocks* and the layers of pooling are called *transition layers*.

An interesting remark is that since DensNet layers have all the previous layer knowledge fed as feature-maps as input it can go by with relatively narrow layers with few channels and still perform state of the art accuracy. This is great because it allows for narrower architecture that results in low computational costs for similar performance. To further up the performance gain some other tricks from He et al. 2016 [30] are used with the 1×1 convolution as a bottleneck layer before 3×3 convolution to reduce the number of input feature maps. This results specially fruitful bearing in mind the amount of inputs each layer has.

All this results an interesting iteration of ResNets that yield similar accuracy with significantly fewer parameters and thus complexity. A side to side comparison of the top-1 error rates on ImageNet can be seen at figure 10.

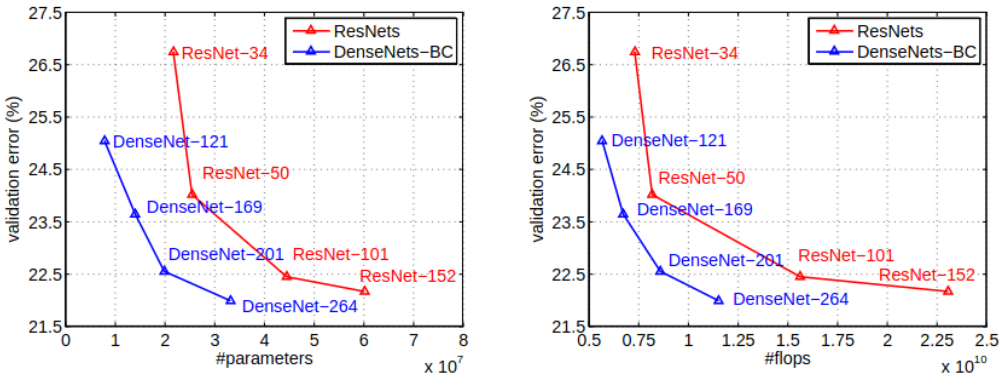


Figure 10: In these charts the more efficient use of parameters can be seen with a curve that gives less validation error given significantly smaller parameter complexity. These charts are extracted from the groundbreaking work by the original DensNet paper[31]. The DensNet implementation particular to the experiment has some further optimizations to better demonstrate the step in performance and simplicity.

5.3.3 EfficientNet

The final model to be presented is the EfficientNet presented by Tan and Le in the 2020 paper [33]. The main idea about this neural network architecture is a rethinking on the scale of the models created. The study focuses on the different scales of the model to balance depth, width and resolution to produce an architecture that properly balances them for the sake of better performance. A performance comparison can be seen on figure 11

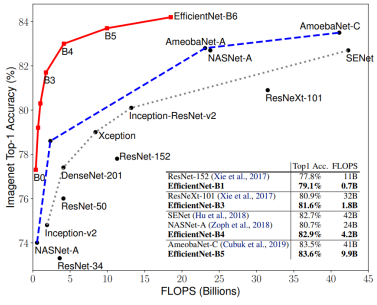


Figure 11: A chart that shows how EfficientNet with its different sizes compares with other models.

Before this model most of the Neural Networks only focused on Depth as the main performance enhancer. So deeper networks meant better performance. However this produces harder to train networks that often don't benefit from all the layers enough. Consequently, width is explored in order to cap-

ture more fine-grained features and making the network easier on training and resolution allows for finer patterns to be captured.

So an observation is made that any scale up in dimension brings some type of benefit with potential drawbacks. Because of this, another observation is made that balancing these three must be key to properly take advantage of all the benefits without triggering any important drawbacks. A compound scaling method is proposed according to a particular coefficient that uniformly scales the network. The method can be described as this:

$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{5}$$

With this baseline guidelines they take architectures like MobileNet or ResNet and iterate with this design factors in mind producing a variety of alternative version of different sizes for each tasks. For our project we will use the smaller B0 version since our image size (resolution) will be relatively small.

5.3.4 Gradient-based Localization

Deep learning models sometime learn in unconventional or unkown ways producing unexpected results or expected results through unexpected or wrongly based patterns or means. Because of that, is highly relevant to have a way of the model to produce a visual explanation for the decisions made within the model to produce the final classification. Furthermore, it can add some degree of value as cheap and moderately useful localization for specialized users that are familiar with the dataset that can take the classification and explanation as a starting point for deeper analysis (for example in the medical field). Visualization might also help with models that appear to classify correctly within certain conditions but fail miserably with some slight tweaks because the learning is not focusing on distinctive enough features.

At first the models that implemented this type of localization used particular architectures in order for the localization to happen. This approach reused most of the network, mainly all the convolutional steps with the trained weights and biases. Changes where done mostly on the last layers before the output where late feature maps could be found. These late feature maps contain abstract features that are relevant for the further *Fully Connected* layers to classify but the information is still mapped in a matrix fashion. This original *class activation maps* (CAM) using global average pooling (GAP). This architecture performed GAP over the last feature maps and connected them to a simple single fully connected layer architecture. Given the simple architecture importance of image regions can be easily identified by projecting back the weighs of the fully connected layer of the output to the feature maps. The architecture can be seen in figure 12.

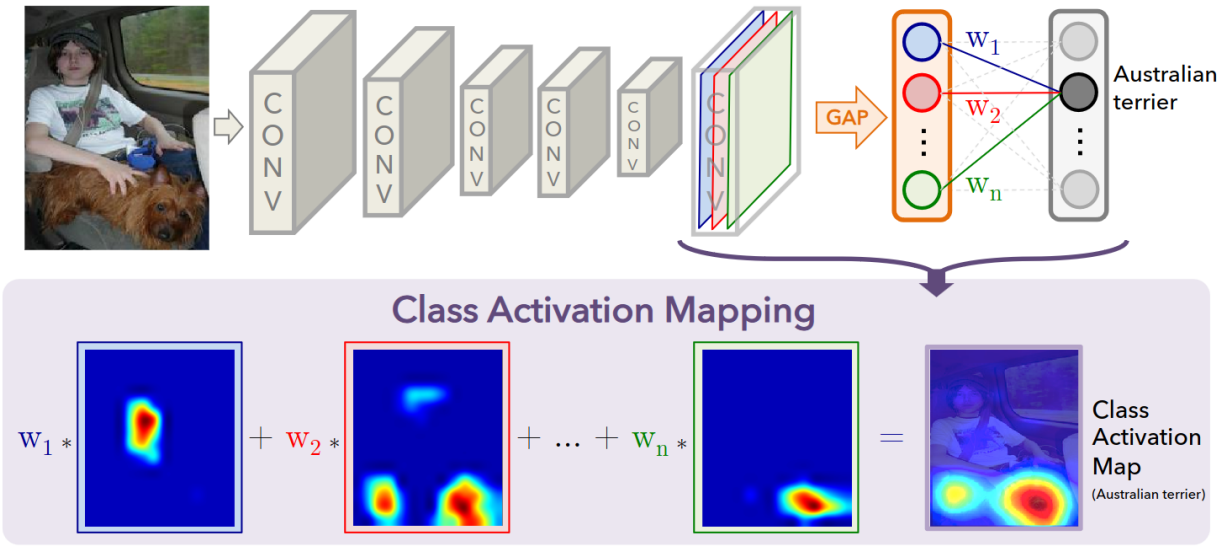


Figure 12: Architecture of the original CAM method extracted from the CAM paper [17]. It can be appreciated that after the conv layers the highly expressive last features are plugged to a new simple architecture of a Fully Connected layers with special weights that are used to creat the Class Activation Mapping.

The resulting heat-maps are defined with the following formulas from the original CAM paper [17].

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y) \tag{6}$$

Where S_c represents the class score of a given class, $f_k(x,y)$ represents given k feature maps the activation functions of the feature maps before GAP, w_k^c represents the weight that relates the feature k to the class c (it's importance), $\sum_{x,y} f_k(x,y)$ represents the result of average pooling for a given k and finally we can compute the $\sum_k w_k^c \sum_{x,y} f_k(x,y)$ global expression

$$M_c(x,y) = \sum_k w_k^c f_k(x,y) \tag{7}$$

Then $M_c(x,y)$ represents the final class activation map coordinates and it equals to $\sum_k w_k^c f_k(x,y)$, the sum of the weights per class multiplied for each pixel of each of the k feature maps.

So basically the Class activation map is a weighted sum of all feature maps according to a classifier weight. This implementation which was originally used for the original chest-xray8 model [5] has a couple compromises in the sense that there's a certain trade-off between localization and explainability and overall performance. For one, you have to substitute the

architecture of the final layer for a simple one layer distribution in order to carry out the calculation, limiting the possible architecture complexity and thus the performance. Some training has to be done upon the base model in order to train the new weighs so they perform properly with the classification task. This problems make it harder to develop multiple base models with the localization idea without significant extra computation and modification of the baseline models. For this reasoning the search for a better model led to the gradient based alternative of cam found in the Grad-CAM paper by Selvaraju et al. [34].

This technique for obtaining visual explanation or CAMs uses gradients for any target class within the data and allows for an alternative way to obtain map highlight of important regions of the image for the particular class prediction. As with previous CAM representations the feature maps used are deep in the network in order to retain both spatial information and high level semantics. Let us look to the algebraic formulas that describe the Grad-CAM definition [34] in order to understand the use of gradients to compute the visual explanation.

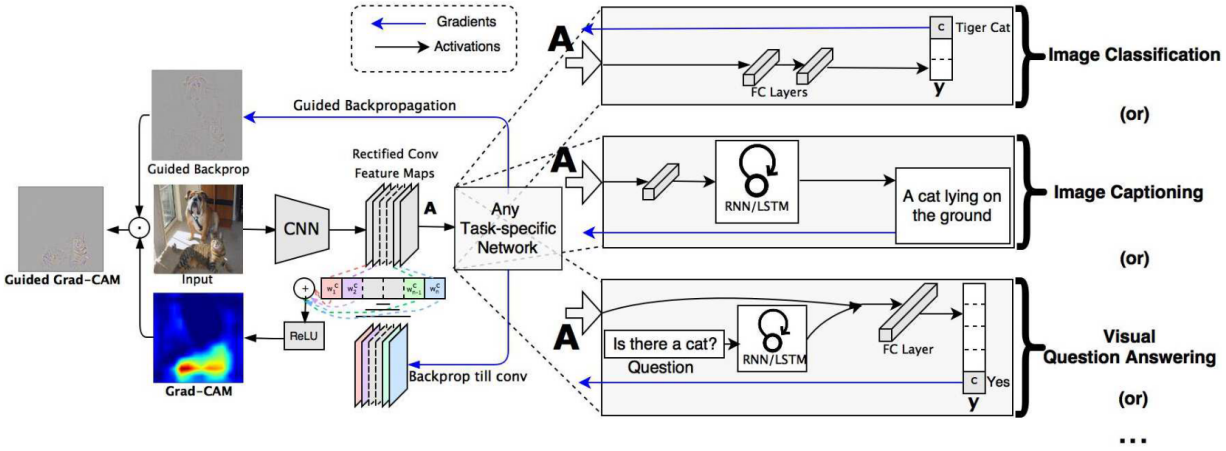


Figure 13: Network architecture extracted from the Grad-Cam original paper [34]. It can be appreciated that after the feature maps the following architecture doesn't really need any specific architecture and can work with multiple different underlying final layer architectures as long as they are mathematically differentiable (so we are able to obtain gradients).

The network architecture can be seen in figure 13. The final localization map is defined as a matrix of $L^c_{\text{Grad-CAM}} \in \mathbb{R}^{u \times v}$ of width u and height v for a given class c. The particular gradient needed to compute the map is the gradient of the class score y^c with respect to the selected feature maps A^k obtaining the gradient $\frac{\partial y_c}{\partial A_k}$ (not to be confused with the gradient of the loss function). Finally all the gradients are put through global average pooling in order to obtain neuron importance weighs similar to the previous class feature weights:

$$\alpha_k^c = \underbrace{\frac{1}{Z}\sum_i\sum_j}_{\text{global average pooling}} \underbrace{\frac{\partial y_c}{\partial A_{k_{ij}}}}_{\text{gradients}}$$

This formula obtains a weight α_k^c that captures the importance of a particular feature map k for a target class c.

Finally as with the previous CAM method a weighted combination is performed with the calculated importance weights along with a ReLU activation function:

$$L^c_{\text{Grad-CAM}} = \text{ReLU} \underbrace{\left(\sum_k \alpha_k^c A_k \right)}_{\text{linear combination}}$$

The use of ReLU is due to the interest in positive influence over the class of interest. This enhances the localization capabilities limiting the information that can be attributed to other classes.

With this new approach most limitations of previous localization techniques are reduced considerably. Since Grad-CAM not only can use architectures as complex as they are but also can be used with no previous training only using the gradients generated with back-propagation with further localization capabilities.

5.4 Pytorch framework

PyTorch is an open source end-to-end machine learning framework that's prepared for fast and flexible experimentation and efficient production through an ecosystem of user-friendly python libraries and tools. Among those we can find some generic tensor libraries like *torch* that have strong GPU support through NVIDIA's own frameworks. Or even some fully fletched neural networks libraries like *torch.nn* that contain many utility functions and predefined layers and loss functions that can be configured for any particular architecture with relative ease and reasonable performance. Finally, some general utility functions can be found in *torch.utils* where DataLoader functions are implemented in order to offer full end-to-end support from the data to the actual predictions[35].

The framework is Python centered and as such deeply integrated into the language. That means that it's designed to be intuitive and easy to use with straightforward debugging without an asynchronous opaque execution engine. Although, it can support C++ to enable research in high performance and low latency applications that favour such languages. Also many engines that the PyTorch framework interacts like *autograd* are written in C++ in order to combat Python overhead.

In order to offer the GPU acceleration and parallelisation it's integrated with NVIDIA's libraries like cuDNN and NCCL to maximize performance. That way all the low level heavy lifting in terms of performance is done by mature and tested neural network backends. Some custom memory allocators for the GPU maintain the networks maximally memory efficient.

6 Implementation

After all Theoretical Background is lain down some remarks on the implementation need to be presented in order to understand the development process and different ensembles created for the project.

6.1 Dataset

The dataset used for the project as briefly introduced in the previous chapters is the ChestX-ray8 provided by the NIH Clinical Center and introduced in the 2017 paper by Wang et al. [5]. The initial dataset is extracted from the official NIH site database[36].

This project aimed to capitalize on the readily available RX scans stored inside the *Picture Archiving and Communication Systems* of Hospitals in order to create a labeled database that could be used to feed deep neural networks. The resulting *Computer-Aided Diagnosis* systems introduced in the paper could harness the information of the labeled images and provide interesting performances with a simplistic low training approach. In this project an iteration of the proposed model will be introduced.

The original dataset is comprised of $108,948$ frontal-view X-ray images of $32,717$ unique patients. The version that is currently being distributed through the original web page[36] has an update that expands upon the original one with some more images rounding to a total of $112,120$. The images where collected between 1992 and 2015 and come with labels obtained through text mining.

Let's further develop the labeling process in order to asses it's credibility and potential compromises. As described previously the labels where mined from the institute *PACS* systems on which radiological reports associated with the images can be found. In order to obtain the final label some Natural Language Processing techniques where applied to detect keywords on the corpus. A special consideration has to be taken since most reports include negation and uncertainty which should not lead to a classification. In the end, an image will have a "Normal" label for no findings or some combination of pathology labels. In order classify the labels a two pass approach is used. Firstly, a pass to detect all disease concepts in the corpus, specially within the findings and impression sections. Secondly, a pass to determine whether it's a normal healthy instance with no findings. A combination of several technologies and tools is used along with some paper original novel rules of negation/uncertainty to account for special cases with multiple subjects. In the end the total precision obtained is about 0.90 and the total recall of 0.91 which are very positive results.

The processing done to the X-ray images a simple resizing from the typical 3000×3000 X-ray image to a 1024×1024 bitmap image without losing detail. Some *Bounding Box* labeling is done by hand by board certified radiologists in order to evaluate the localization performance.

6.1.1 Metadata and Labels

In order to understand the images within the dataset we will take a look into the associated metadata and prediction labels that the dataset offers.

The official *README* file[37] introduces the metadata of each image as such:

- Image Index *(int)*: Corresponds to the filename of the image that the metadata entry represents, it's codified with an eight digit number that codifies a particular patient followed by a three digit number that signifies the chronological order of the images within the same patient
- Finding Labels *(str)*: This is a string that contains the labels that are positive for a given image with multi-label encoding with several layers separated with "—".
- Follow-up # *(int)*: A numerical variable that contains 0 if this image is the first one of the chronology and a number n that represents how many images is this image a follow up to.
- Patient ID *(int)*: An anonymous and abstract patient id that allows for the researcher to tell which images are from the same patient without disclosing any extra information.
- Patient Age *(int)*: One of the only variables that discloses the minimum personal information possible that is relevant to the medical condition of the patient. Age could be added as an extra feature.
- Patient Gender *Categorical (F,M)*: Gender of the patient might be relevant to train the model correctly to different body features associated with gender or to draw statistics from the models performance associated with gender.
- View Position *Categorical(AP,PA)*: View position can be anterior-to-posterior (AP) or posterior-to-anterior (PA) depending on which structures of the human body (front or back) are closer to the sensor when taking the RX. This might be of some relevance since the more highlighted organs will be the closer to the RX machine.
- Original Image Size *(int,int)*: Original image size in width and height.
- Original Image Pixel Spacing *(int,int)*: Original pixel spacing within the original image size.

The dataset also comes with a small bounding box coverage for approximately 1000 images and a suggested data split taking into account the patient id for the split.

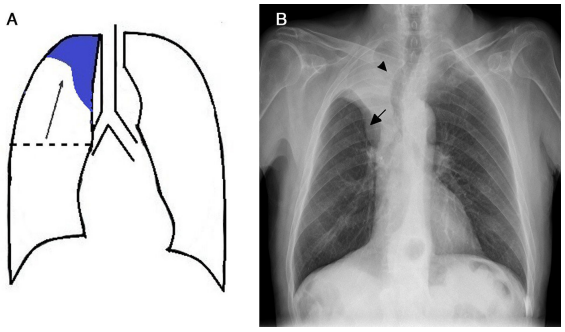
6.1.2 Basic Medical Explanation of Labels

Since some degree of localization will be introduced in our model, it is highly important that some study of the common diseases that are labeled is done in order to understand if the localization is properly done. Furthermore knowing what to expect is relevant in order to evaluate the explainability of the model. Since, if a particular label is decided from a non relevant part of the image this would signify deeper model problems and potential errors. Most of the definitions are directly extracted from sources related with the National Library of Medicine of the NIH [38].

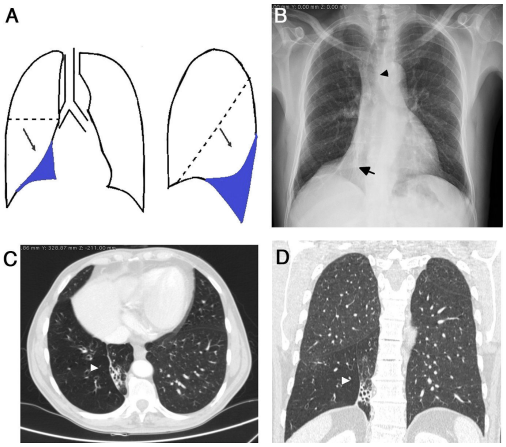
Pulmonary **Atelectasis** is characterized by a partial or complete collapse of sections or the entirety of the lung which results in reduced gas exchange and respiratory function. In particular, the collapse happens in small airways where the exchange of blood's CO_2 and the O_2 present in air is cut causing an intrapulmonary shunt, since blood meets the alveolars but air doesn't. This is particularly common among patients undergoing general anesthesia or prolonged bed rest. [39].

In terms of evaluation with x-ray sometimes atelectasis can be overlooked if the condition is not advanced. As described in the Woodring et al. 1996 paper the direct signs of atelectasis are crowded pulmonary vessels, crowded air bronchograms, and displacement of the interlobar fissures. Indirect signs of atelectasis are pulmonary opacification; elevation of the diaphragm; shift of the trachea and heart; and approximation of the ribs[40]. In our case the indirect signs are the ones to look for in the RX scans. Mainly, loss of volume and an increase in opacity along the affected region.

There's a vast array of different configurations on which the collapse can happen but a selection of two very different visual examples has been chosen from an article on the topic by Cortés et al. 2014 [41] shown at figure 14:



(a) This example contains the collapse of the upper right lung.The misplacement of the trachea is also of relevance for the diagnostic.



(b) This other example shows the collapse of the lower right lung with also some trachea misplacement to the right.

Figure 14: Two cases of Atelectasis extracted from the cortes et al. paper [41]

Consolidation is defined as air-space shadowing with other abnormal denser materials that produce slight opacities. This can happen due to infection like pneumonia or cancer[42]. It constitutes a sign of other pathology more than a particular illness itself since the material that fills the space can look visually similar but be of vastly different nature. Visually the opacities look more diffuse than other signs like effusion or mass which are fairly opaque.

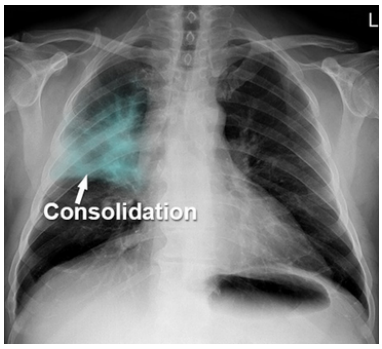
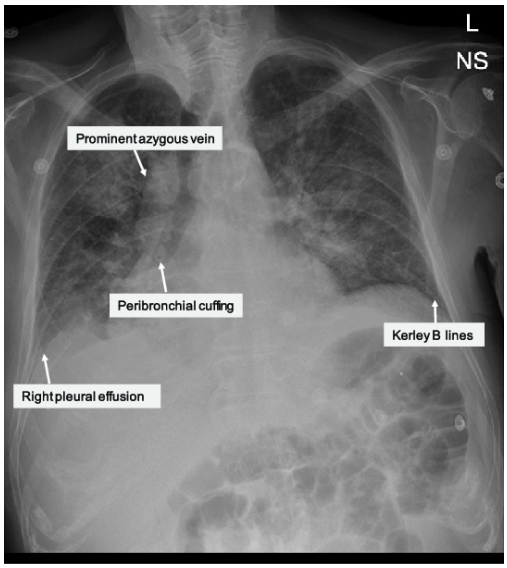


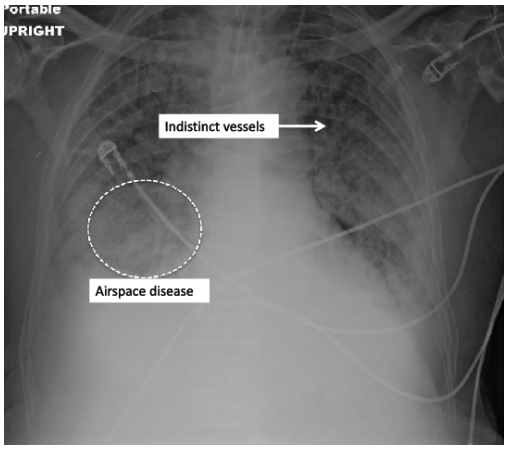
Figure 15: An RX example of consolidation on the right lung highlighted with blue and extracted from [42].

Pulmonary **Edema** can be defined as an excessive accumulation of extravascular lung water. The result of this accumulation is an impairment of the respiratory gas exchange, resulting in respiratory distress and often the need for mechanical ventilation. This can be a result of chardigenic problems (related to cardiomegaly) or a lymphatic system's inability to clear lymphatic fluid among others[43].

The visual evaluation of a pulmonary edema in terms of chest radiography present the following graphical cues: peribronchial cuffing, indistinct vessels, and septal (Kerley) lines as it can be seen on the figure 16a. In distinction, alveolar features present with patchy opacities, airspace disease seen in figure 16b. In general features tend to happen within the affected lung, around the mid to lower lung.



(a) A chest X-ray that shows the features of cardiogenic pulmonary edema.



(b) A chest X-ray that shows the features of Acute respiratory distress syndrome as in an example non chardiogenic edema.

Figure 16: Two cases of edema extracted from Assad s, et al. 2017 paper [43].

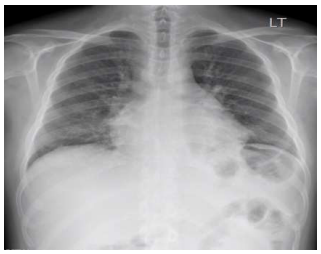


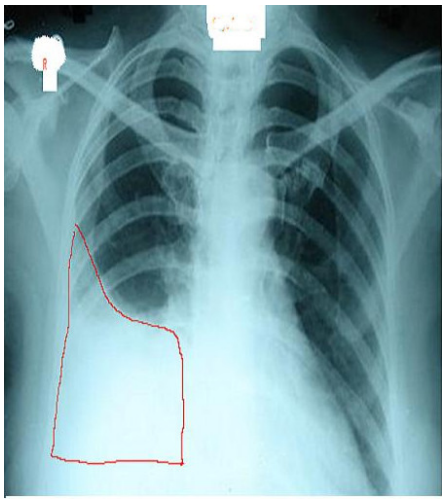
Figure 17: An RX example of cardiomegaly derived from a complex medical condition presented on a case study [44].

Cardiomegaly corresponds to an umbrella term for various conditions that produce heart enlargement which remains normally undiagnosed until late symptoms. The definition states that a 50% increase of the cardiac silhouette is classified as cardiomegaly. Many diseases are related to this condition which is an alarm signal that leads to further investigation of the source problem since the condition has high risks and shows high mortality[45].

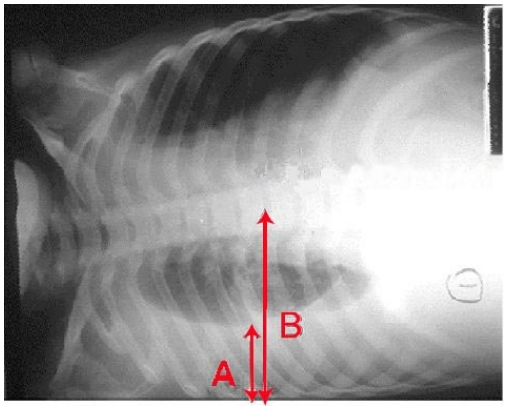
In terms of evaluation it is fairly straight forward since the focus of the diagnostic is evidently the cardiac area. The enlargement can be due to any particular ventricle which produces different deformations.

Pleural **Effusion** is the abnormal accumulation of fluid within the pleural space, which is the thin cavity between the layers surrounding the lungs. This affliction can have various etilogoies that are related with other labels within the set. Particularly heart failure (which can be represented as cardiomegaly, lung masses and pneumonia.

For this pathology the chest X-ray is the main evaluative method which is promising for good performance on the particular label. The main visual cue is the accumulation of liquid that creates opacity on a particular pattern. Since it's accumulated within the membranes the liquid can move through the cavity by it's own weight and it is found in the bottom if done upright and on the side if the RX is taken in a lateral view.



(a) A chest X-ray that shows the accumulation of liquid on an upright RX positive of effusion.



(b) A chest X-ray that shows the accumulation of liquid on an lateral RX positive of effusion.

Figure 18: Two cases of effusion extracted from Krishna R, et al. 2024 entry of StatPearl Publishing [46].

Pulmonary **Emphysema** is a progressive lung disease that falls under the Chronic Obstructive Pulmonary Disease (COPD) umbrella. It is defined as a common an treatable disease characterized by persistent respiratory symptoms and airflow limitation that is due to airway or alveolar abnormalities caused by exposure to toxic gases or microparticles. It is characterized by abnormal permanent enlargement of lung air spaces with the destruction of their walls and tissue with loss of elasticity[47].

Since the affliction is a COPD and it's progressive the main way of diagnosis is Pulmonary Function Testing in order to measure the capacity of the organ and thus the severity of the illness. A chest X-ray diagnosis is useful but in very extreme cases. Destruction of alveoli and air trapping causes hyperinflation of the lungs with flattening of the diaphragm, and the heart appears elongated and tubular[47].

Cystic **Fibrosis** is chronic illness caused by a genetic mutation in a gene that codes protein trans membrane conductance regulator. This leads to a decreased secretion of chloride and consequently increased absorption of sodium into the cellular space. This sodium abundance leads to increased absorption of water which results in thicker mucus which can lead to mucous plugging with obstruction. Thus commonly producing sinopulmonary infection and pancreatic insufficiency[48].

In terms of the lungs in particular plugging occurs at the brochioles resulting in obstructive clinical disease. Due to obstruction bacterial growth is propitiated and more mucus and inflammation is produced creating a positive cycle. Since it's a genetic mutation and it has an impact on chloride secretion genetic the first step is sweat test and then DNA to diagnose[48]. In terms of X-ray the main signals to look for are signs of infection and inflammation as well as other labels within the dataset like atelectasis.

A diaphragmatic **Hernia** is a condition where abdominal contents protrude into the toracic cavity due to a physical defect to the diaphragm. If acquired via trauma it is variable in it's symptoms varying from mild to life-threatening. Mostly it's a congenital feature of neonates[49].

In the case of x-ray scans they are of value for postnatal congenital and acquired cases. The RX have some variety since it depends on the location, the size of the defect, the abdominal organs involved and the displacements of the organs. It can be paired with pleural effusions from the defects. As such the X-ray cues are basically along the diaphragm and can relate to other labels.

Figure 19: RX scan of a Acquired Diaphragmatic Hernia located in the lower left lung. Contributed by S Bhimji, MD on [49].

A solitary pulmonary **Nodule** (SPN) represents a discrete, rounded opacity within the lung parenchyma that is < 3 cm in diameter and completely surrounded by lung parenchyma without associated lymphadenopathy, atelectasis, or pneumonia. These nodules are frequently incidentally detected on imaging studies performed for reasons unrelated to pulmonary disease. Even though most cases are benign, it is essential to determine the underlying cause because lung cancer is the leading cause of oncological death in the United States[51].

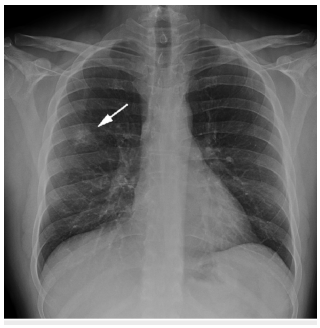
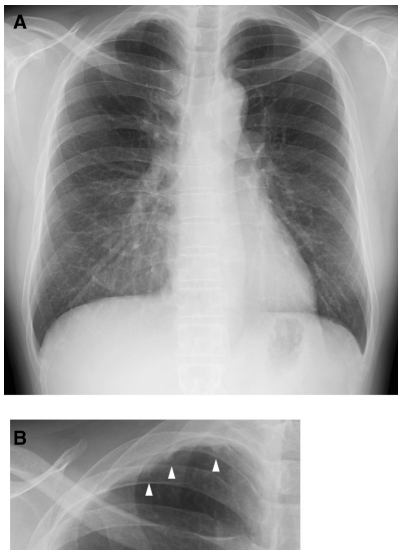


Figure 20: RX scan of a solitary pulmonary nodule highlited with a white arrow [50].

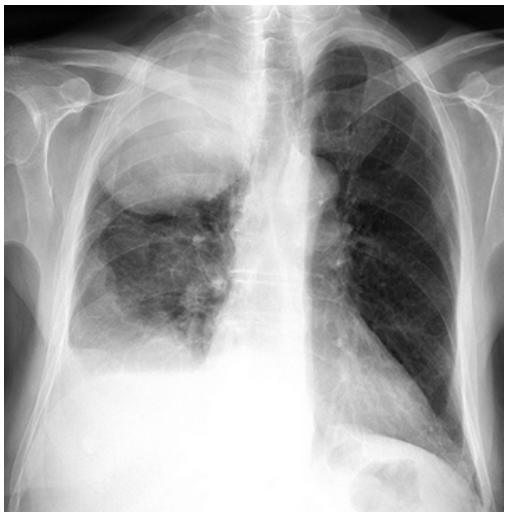
Chest x-ray (CXR) detects a significant amount due to being an inexpensive rutinary check even tough it's not the best method. The visual cue is a well defined rounded object on an otherwise perfectly normal scan.

Lung **Mass** is basically a development of a nodule to a large mass. It features the same opaque presence of a nodule but with greater size than the usual < 3 cm size. Nodules often could mean no harm to the patient but masses are a greater indicator of lung cancer and should be studied as such[42].

Pleural Thickening is a common finding within X-ray testing created from a fibroelastic scar involving pleura and lung tissue. It usually involves the apex of the lung, in the so called 'pulmonary apical cap' as a non pathology related event or it can signify some other pulmonary diseases like bacterial infection, lung cancer(from asbestos inhaling), and idiopathic interstitial pneumonia[52]. As such, visually it can be highlighted along the border of the lung on the apex part or in the sides.



(a) RX scan of a pleural thickening case in the apex of the lung [52].



(b) RX scan with mass on the apex of the right lung and an effusion extracted from [42].

Figure 21: Examples of pleural thickening and mass.

Pneumonia is an umbrella term for a group of illnesses caused by live organisms that result in infection of the lung tissue. Depending on the organisms responsible the treatment and severity of the illness varies. As such it can be obtained by bacterial, viral and fungal means. It typically carries all the infectious traits of other types of infections like inflammation and fever with the added complexity of happening in the lungs creating excess mucus that can lead to respiratory difficulties[53].

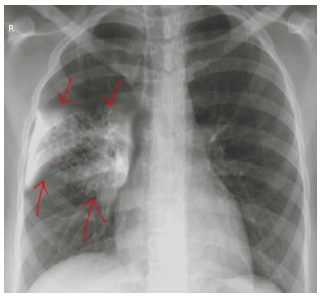


Figure 22: Chest radiograph demonstrating alveolar infiltrates in aspiration pneumonia. Contributed by O Chaigasame [53].

In terms of detection this mucus and inflammation are the main cues for finding pneumonia in the preferred clinical diagnosis method, the RX scan. As such it is a perfect label since the main diagnosis method is the one at study.

A **Pneumothorax** is a collection of air outside the lung but within the pleural cavity. It occurs when air accumulates between the parietal and visceral pleurae inside the chest. The air accumulation can apply pressure on the lung and make it collapse[54].

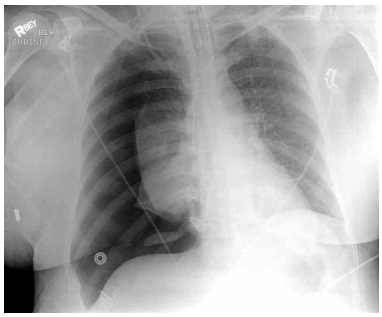


Figure 23: Right Pneumothorax Radiograph. This image shows absent lung markings on the lateral side of the right pleural cavity and a collapsed right lung. Contributed by S Dulebohn, MD [54].

Radiographic findings of 2.5 cm air space are equivalent to a 30% pneumothorax. Occult pneumothoraces may be diagnosed by CT but are usually clinically insignificant.

Finally, **Infiltration** is the result of a substance denser than air (like in consolidations) but it can settle anywhere within the lung tissue not limited to air spaces. This leads to slowly resolving pneumonias that are hard to detect for absence of symptoms until heavily developed. Thus the visual cues are similar to consolidation but with a ore broad potential opacity pattern that occupies not only airspaces but the whole lung[55].

6.1.3 Statistic study

Some statistic study was undergone in order to properly visualize the data so that the dataset could be understood to a greater extent.

In terms of **age** we can observe that naturally an older section of the population is represented within the dataset. This makes sense in terms of the previous disease exploration since, except some foreign body incidents with children, pathology related to pulmonary illness have some correlation to lifestyle effects that manifest in later stages of life like smoking or toxic exposure. Apart from aging which sometimes can create problems of their own. The mean of the age is 46.59 with a standard deviation of 16.55 which reinforces the previously stated theories. A peak within the violin plot 24 can be seen close to 60 which could be attributed both to enhanced clinical presence of a afflicted group of the population or a simple population trends. No particular difference is spotted between the male and female in terms of age distribution. Male label is more common with a frequency of around 56% leaving the other 44% for the Female label. Thus, there is not a lot of **gender** imbalance that could produce a model that performs better on certain gendered patients.

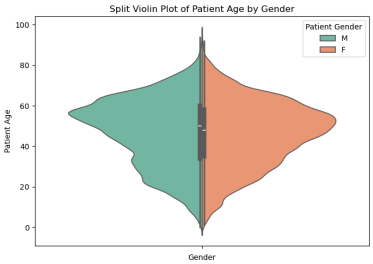
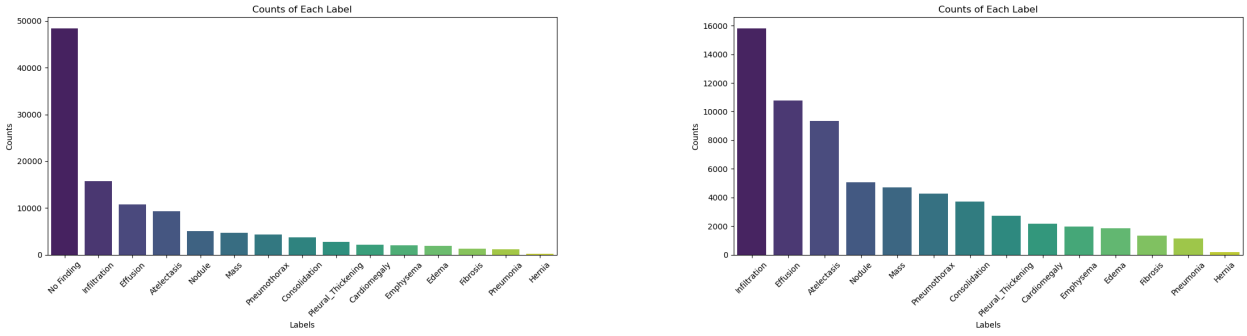


Figure 24: A violin plot that represents age on the y axis with hue for Patient Gender. Generated by the author with seaborn.

Now in terms of **counts for each label** it can be appreciated that the main label within the multi-label findings is "No Finding" on the barplot 25a. Because of that a variety of models from different sources tend to opt out no findings to boost the learning of interesting features for the proper diagnostic of the positive labels. Some minor experimentation should be done on the authors behalf to test if this has a positive impact on classification of positive labels a part from the obvious performance gain of processing almost fifty thousand less images per epoch. When taking out "No finding" label from the count we can see that representation from the classes is still uneven but better than before 25b. This hints that some studies on potential data modification for equal representation would be of great use to better understand under represented classes.



(a) No Finding is too common.

(b) Better visualization without No Finding.

Figure 25: Two barplots for multi-label counts generated by the author with seaborn.

In order to see the actual counts of pathology presence on the patient label we count how many affected patients for each pathology there are within the dataset in the barlpot 26. This helps to grasp which labels are more common among the patients rather than in the dataset. Since a single patient can produce several positive labels that account for the same illness. We can see that some changes happen between the general count and the patient count. Infiltration still is the most common with six thousand plus cases but the latter that follow are in a different order.

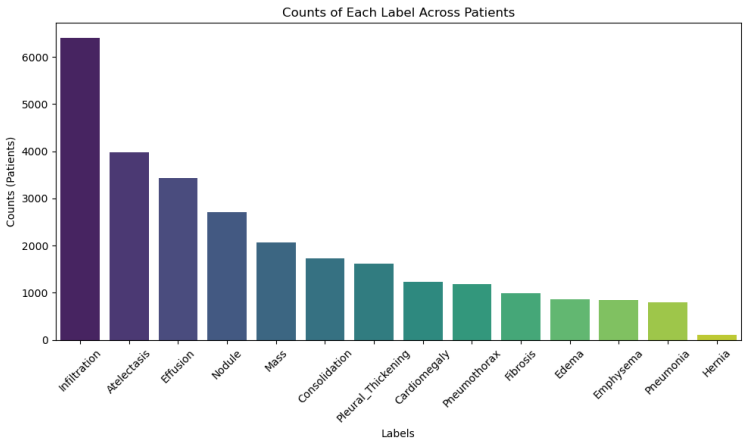


Figure 26: A barplot that depicts the patient count for each illness to better understand how common illnesses are on the patient label. Some changes in the barplot order happen due to this. The barplot was generated by the author with seaborn.

Now looking at the **correlation** between the different variables within the set we can see some interesting correlation data between the different labels that further expands our knowledge on the previous illness study.

At first, all numerical variables where considered within the correlation study 27 but there was little correlation between the labels and other metadata. Mainly, an interesting and unexpected positive correlation with "follow up #" and some labels like Effusion, Infiltration, Consolidation and Pneumothorax seem to hint that those illnesses produce closer monitoring and follow up RX scans. Even though this might also have to do with abundance of the particular labels so it's not very conclusive. In terms of age there are no highlights in terms of correlation with labels and image size and pixel spacing are closely correlated with each other but not with the labels. Because of that the main correlation findings are within the correlation between labels so the focus of the study will be on those labels.

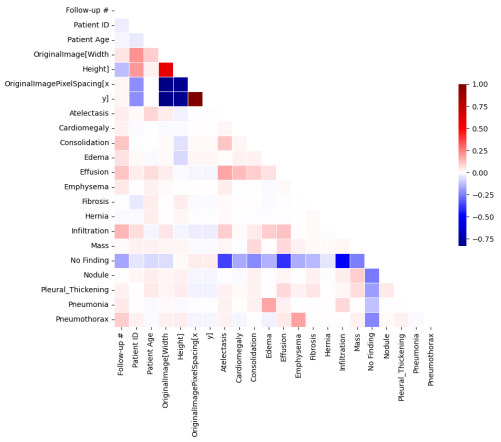


Figure 27: Correlation heatmap for all numerical features in metadata generated by the author with seaborn.

In terms of **label correlation** at first glance we can see in the heatmap 28 that evidently No finding label is very negatively correlated with all the finding labels which is to be expected since the absence of positive labels produces the negative one. On a more interesting note we can corroborate that **Effusion** is closely related to many other labels since the abnormal accumulation of fluid within the pleural space can have many etiologies as it was described in the medical exploration. Effusion mainly positively correlates to Atelectasis, Cardiomegaly, Infiltration, Masses and Pleural Thikening. Some correlations between labels can be attributed to how similar some definitions are. Since **Effusion**, **Edema**, **Infiltration** and **Consolidation** are all related to the presence of liquid or dense materials in different lung spaces some of them can overlap when a particular pathology is happening. **Nodule** and **Mass** are related too since Mass accounts for a developed significantly sized Nodule. **Hernia** is not related to anything in particular since the pathology does not typically produce many of the other labels. Furthermore there are not a lot of cases so lack of correlation can come because of under representation. One of the most positive correlations is **Pneumonia** and **Edema** which are related because of a close relationship with mucus trapped within the lungs and the impaired respiratory gas exchange characteristic of Edema. Finally, **Pneumothorax** and **Emphysema** correlate positively since Emphysema is characterized by enlarged lung air spaces with eventual destruction of the tissue that leads to Pneumothorax. Furthermore, Emphysema is typically diagnosed with Pulmonary Function Testing and when it's diagnosed through RX scan the case is often extreme and thus has presence of Pneumothorax.

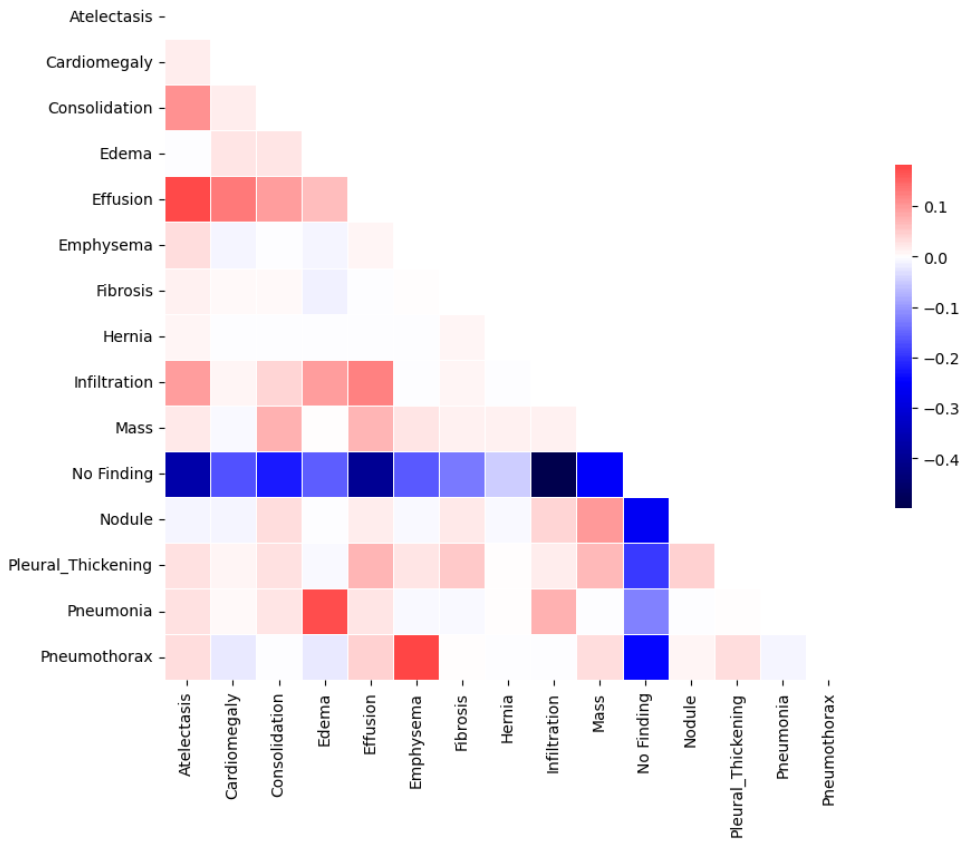


Figure 28: Original correlation heatmap for multi-label class features created with seaborn.

In order to futher understand the possible clusters within the data **PCA** analysis was done to reduce dimensionality of relevant data in order to try and foresee possible trends of the model once trained.

Non-useful variables like id or image size are taken out of the PCA study. Since we are looking for clusters that are related through medical related features like the labels or patients age and gender trying to look for trends. Since PCA only works for continuous type variables later some better suited methods for clustering where tried in order to asses some facts about the data. The projections that PCA gives are suited to maximize variance but with binary data lacks meaningful variance and as such it doesn't favour the model[56]. For this reason the resulting graphs are not satisfactory at all. The explained variance by principal components is around 90% at 14 principal components which demonstrates that principal components are having a hard time reducing dimensionality. The plots that visualize all data points within the PCA projection is not very illustrative but the biplot allows us to visualize how the original features lay in the projection in figure 29. In that sense we have very low variance explained, so conclusions are few but we can see how pc1 seems to capture the variability between healthy patients towards the positive values and ill patients on the negative region. Overall we can see a representation of the correlations that we saw earlier with the angle of the variables.

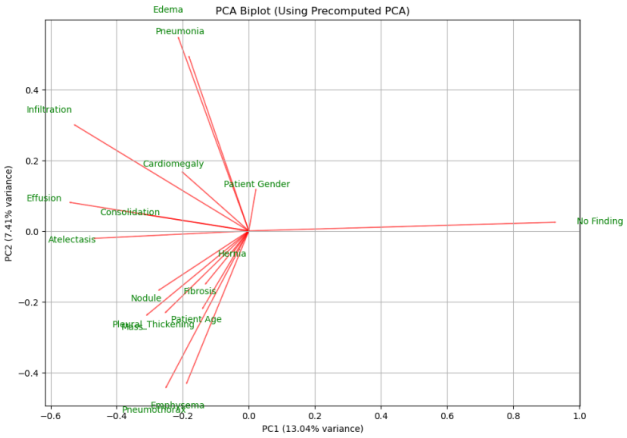


Figure 29: PCA biplot with original variable vectors generated with matplotlib.

In terms of other possible methods we use Van der Maaten and Hinton **t-SNE** method[57], to visualize the data points in a less crowded manner as it was in PCA with two dimensions. This method is better at capturing nonlinear features since it relies on preserving the local structure of the data maintaining all the complex patterns of the data. The only drawback is that complexity of t-SNE is $O(n^2)$ which is great for the type of problem it's solving but bearing in mind the size of the metadata it's a significant cost.

As we can see in figure 30 apparitions of determinate classes are locally close and the most common classes revolve around the outer parts with clear clusters. Some closely related diagnostics like Atelectasis and Effusion are in a close cluster individually and when they appear at the same time bearing some similarities. Mass and Nodules are also clustered together, as well as Pneumothorax and Empysema. Consolidation, Edema and Cardiomegaly are also arranged in similar clusters. In general all previously expected illnesses that where closely related are well captured by the clusters generated within the t-SNE representation. Not all data points where plot since there are a lot of classes and combinations of them and it's not feasible to represent all of them in one singular space but the most common classes in occurrences where the ones shown in the plot.

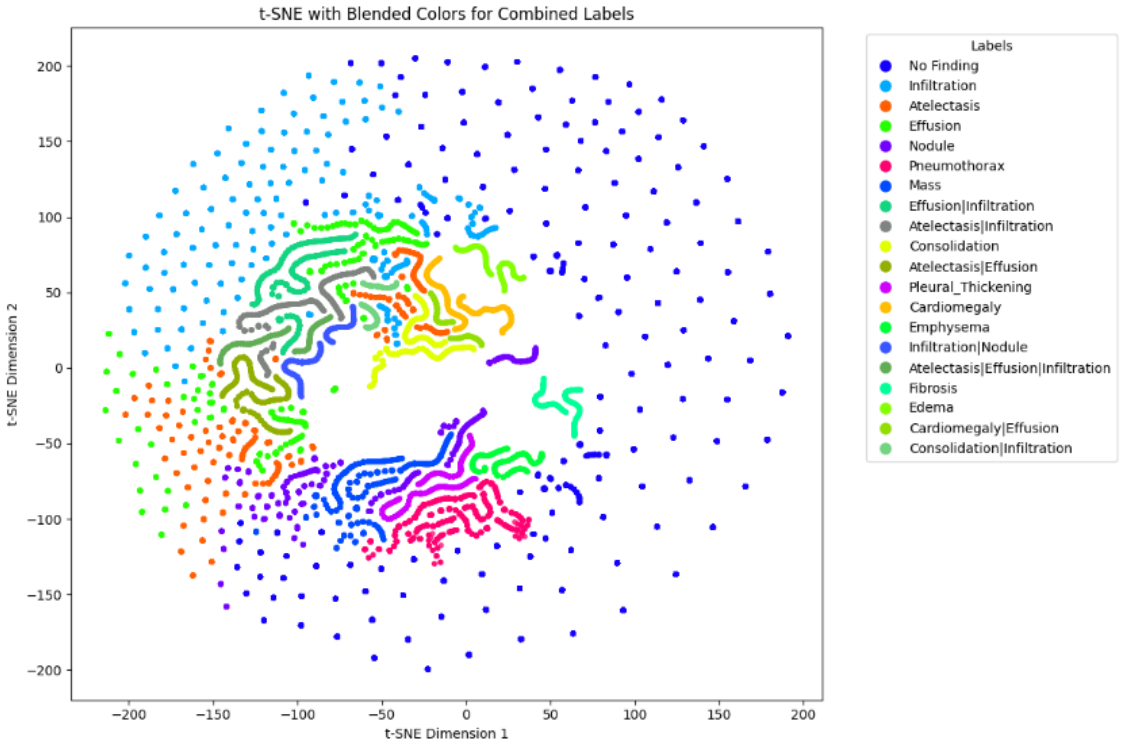


Figure 30: T-SNE two dimensional representation created with sklearn tsne and matplotlib.

Previously it was mentioned that some bounding boxes created by radiologists are facilitated along the dataset and in order to check and grasp the potential localization of different pathologies a **heatmap of the bounding boxes** was made in order to see possible tendencies on the different localizations. Only the original 8 labels from the dataset are included so not all the labels are represented but it's still valuable material. In that sense we can take a look at figure 31 and see that there are some very clear tendencies to be studied. Atelectasis is a bit general since the collapse can happen in a variety of places as previously stated. On the other hand, Cardiomegaly is one of the more clear diagnostics with a foolproof centered bounding box all of the time that covers the cardiac region and some of the left lung. Effusion is another diffuse label in terms of visualization the liquid on the pleural space can be spotted along the whole lung. The same goes for infiltration and pneumonia that produce similar visual cues along the lungs in the form of opacities. Masses on the other hand have a tendency to appear on the right lung specially with possible appearances in other parts in a more sparse and localized manner. Nodules in particular ares mall bounding boxes without a clear pattern other than the size. Finally Pneumothorax is more common in the top part of the lungs in the form of a small collapsed part of the lung.

Finally talking about **co-occurrence** in figure 32 we can see similar visual cues as with the correlation with a very straightforward representation that shows classes that are more tightly related with big edges between them. We can highlight Effusion, Infiltration and Atelectasis, which not only are the most common occurrences but they share a lot of co-occurrence with several other labels.

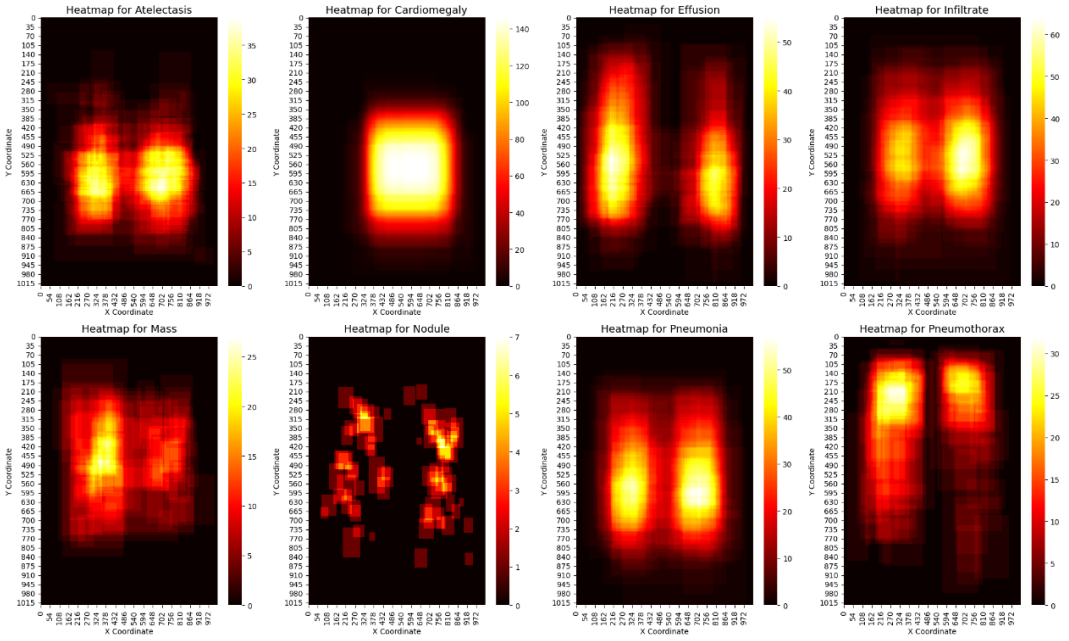


Figure 31: A heatmap of the bounding boxes of the metadata generated with seaborn.

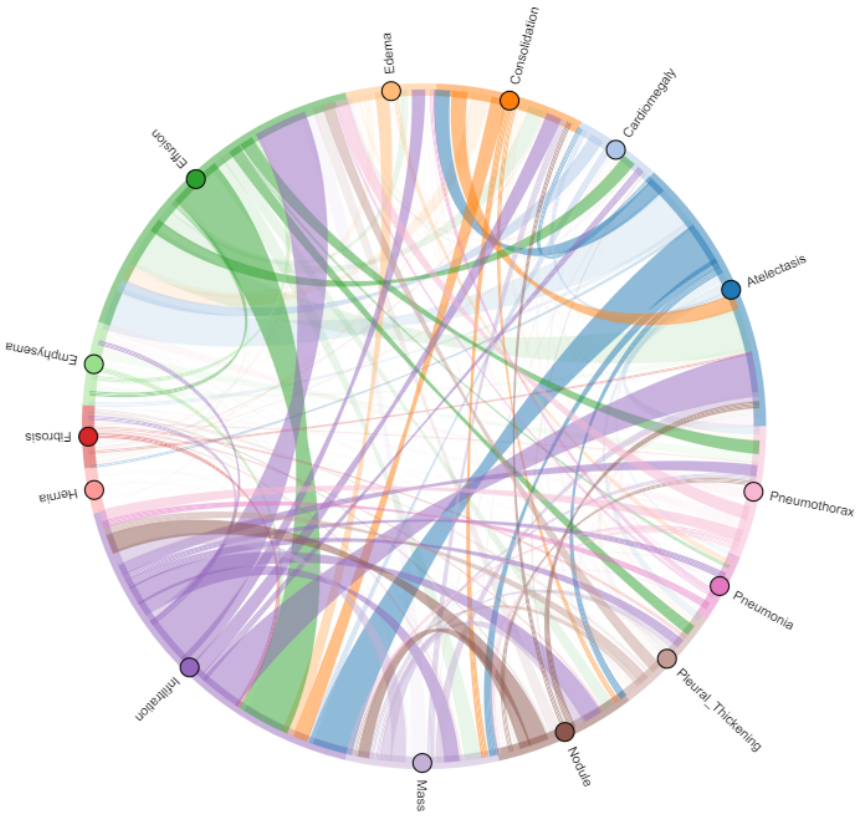


Figure 32: Co-Ocurrence Chord diagram of Positive Labels created with a holoviews extension called bokeh.

6.1.4 Dataset treatment - Preprocessing

In terms of preprocessing the project took the original metadata from the NIH dataset described in the metadata and labels section and generated particular *".csv"* that suit the particular dataset implementation that is to be used for this project.

Metadata is extracted from the original tabular data from the dataset at NIH [36]. On the untreated table, *"Finding Labels"* are codified in a string that has a list of labels separated by vertical bars between instances of labels. Because of that, the labels have to be parsed into individual labels that contain 0 for negative and 1 for positive presence. In that sense we take the decision of creating the *"No Finding"* label to be able to balance the set and asses predictions better. After that all other metadata is discarded apart from the Patient ID which is necessary to properly generate the splits on the data. This is to prevent data leakage. Since images from the same patient being split between sets would defeat the purpose of the data split as separate sets for validation and evaluation. This approach is taken in most solutions for the dataset (like the study of Holste et al 2022 [15]). We perform this with the *GroupShuffleSplit* from sklearn[58] grouping by *Patient ID*.

After that some light work is done to account for class imbalance specially among the *No Finding* label that is very common taking more than half of the cases. With class imbalance one can try to approach the problem at the data-level trying to find sampling methods to make the data more balanced. These where mostly extracted from the paper of Van Hulse et al. 2007[59] that compiled most of the possible methods like Random Under Sampling (RUS) that is simple and appropriate to our case. The main problem is that some of the other more complex methods like Synthetic Minority Over-sampling Technique (SMOTE)[60] are not specifically designed to either high dimensional data like images or multi-label types of dataset. This inability for some class imbalance mitigation methods to be unable to help in multi-label datasets will be a common theme along the implementation. Because of that random undersample was implemented to reduce the overly represented label of "No Finding". Other labels where considered but since this won't be the only imbalance mitigation method of the whole project so it's a good compromise.

Finally all data is stored as *csv* for the sake of the experiments being reproducible and for ease of programming for loaders and datasets.

6.2 Platform

Now some remarks on the selected platform and environments of development will be explained for the sake of reproducibility of the project's code. At first most testing was intended to be done on Boada but some of the foreseen risks and circumstances that eventually took place limited the amount of work that could be put in the environment. Because of that a segment of the experimentation had to be done on a local machine with comparable power and similar testing environment. In the end, the Boada platform was discarded altogether for the reasons stated in the methodology section 4.1.2 so platform environment setup covered will be done exclusively for the local PC.

6.2.1 Environment Setup

As stated in the Practical Methodology section both environments have similar hardware and for extra compatibility in initial states where both where planned to be used some considerations where taken into the local setup to imitate the remote available libraries, CUDA toolkit and Python version.

In the end, even tough the local PC was the only environment used this considerations within the initial parts of the project left a very desirable environment that could be exported easily to other architectures limiting the problems of compatibility. This was achieved creating a Conda environment from the Anaconda Hub. As described in the original website: "Anaconda is an Open-source package and environment management system that runs on Windows, macOS, and Linux. Install, run, and update packages and their dependencies."[61]. Thus, with this environment management tool a particular environment can be created that works across many different OS and particular hardware architecture configurations as long as conda is installed. Conveniently, it offers a special format *.yml* that encapsulates all the particular configurations of a given environment and it can be easily replicated just executing a single command with the file as an input. It can install different versions of python and a variety of particular drivers and libraries specific to the development process so that reproducibility is at it's maximum expression. Since any other researchers can pick up the code, create the conda environment and run seamlessly with no compatibility issues.

The main packages and versions can be seen in table 1. There are many installed packages in the *.yml* file because of dependencies but the main installs are the depicted. Mainly we use *torch* and *scikitlearn* for deep learning functionalities, some quality of life libraries like progress bars with *tqdm* and dealing with calculations like *numpy* and *pandas*.

Table 1: Library Versions from environment.yml

Library	Version
`torch`	2.4.1
`torchvision`	0.19.1
`scikit-learn`	1.5.1
`numpy`	1.26.4
`pandas`	2.2.2
`matplotlib`	3.9.2
`seaborn`	0.13.2
`Pillow`	10.4.0
`OpenCV`	4.10.0
`tqdm`	4.66.5

6.3 Model Architecture

Since all the background pieces and necessary elements for the model have already been lain down we can take a deeper look on the models and surrounding architectures that will be portrayed within the project's code and experiments.

6.3.1 Previous solutions

Some initial remarks on previous solutions where already presented on the initial sections of the thesis but some deeper remarks on the models and ensembles that are currently being used are necessary to select the desired technologies for the project's selected model. A reasoning on why some elements among the different solutions will be taken into account for the suggested solution will be developed too.

- ChestX-ray8 Wang et al. 2017 [62]: Offers a simple solution with the existing models at the time (ResNet50, VGGNet-16) and CAM [17]. This implementation is particularly rudimentary because is just a proof of concept from the dataset creators to demonstrate the performance that could be harnessed feeding the dataset to an array of different state of the art models at the time. The main hurdle a part from using sub-par models like VGG-16 that were a little outdated at the time by models such as DenseNet[31]. Also the use of the original cam without the gradient version required retraining and reduced the expressive power of the last layers being a single fully connected layer. This layered the ground work for what the network should do and later papers worked from here onward.

- CheXNet Rajpurkar et al. 2017 [10]: Offers a slightly developed solution using DenseNet-121 in order to have a deeper network architecture that could learn more from the dataset. Training is relatively straightforward with some considerations on *weighted cross entropy* functions and the *Adam*[63] optimizer which did gradient based optimization on which each parameter has it's own learning rate calculated from estimates generated by the moments of the gradients. It still uses the old CAM[17]. The main point or the paper a part from setting a new state of the art was comparing the model to actual radiologists on tasks, on which the model performed similarly to them in classification metrics. It's mainly single label but they generalize the model for multi-label.

- CheXClusion Sayyed et al. 2020 [11]: Examines the current state of the art in terms of fairness across the different protected attributes (personal data like sex, age or race). In terms of classification it's not too special but it's a more recent implementation of the state of the art. It's very similar to CheXNet with some modifications like the *lr scheduler hyperparameters* and and some *data augmentation*.

- Multi-Objective Evolutionary Design by Lu et al. 2019 [14]: has an automated network design process that falls out of scope since the thesis author has never used neural networks before and jumping straight to such advanced techniques could be a limiting factor on the learning process of the thesis. Since before automating one must understand the basic process.

- SynthEnsemble Achraf et al. 2023 [64]: creates hybrid models between CNN and Transformers since nowdays transformers have developed rapidly. In terms of the scope of the project, this type of model falls out of it similarly to the previous one due to transformers being a far fetch architecture that is vastly different and would not be feasible to do within the 18 ECTS restriction of time. Deeper understanding of more traditional methods was prioritized.

- Long-Tailed Classification Holste et al. 2022 [15]: this project puts a really interesting perspective on the problem that not many people have tackled this deeply and it was the most defining papers in terms of study aspects for this thesis. The main problem is that the technologies described in the paper are mainly designed for multi-class (one-hot encoded) classification which is not what the problem definition was. So from this imbalance treatment from the Long Tailed Classification some studies where undertaken in order to find decent solutions for the Multi-label problem definition.

- Asymmetric loss for multi-label Classification Ben-Baruch et al. 2021 [20]: It's a multi-label loss function that takes into account the imbalance between negative and positive counts in labels around the optimization process prioritizing positive labels that are harder to classify. This loss function is proposed in a general problem like MS-COCO so we want to test how it works with our dataset.

- EfficientNet Tan et al. 2019 [33] creates a new ConvNet architecture appropriately scaled in width, depth and resolution to maximize learning with a compound scale with all three variables in order to extract as much from all. This dimensions are coordinated with a particular ratio and thus the model has several configurations according to the ratio. With it's smaller counterparts being great for prototyping and bigger counterparts that can get that extra performance for a higher parameter count.

6.3.2 Suggested solution

Now that the state of the art is properly defined our initial model can be can be properly introduced. Models to be selected are ResNet50[30] for it's simple yet powerful breakthrough capabilities (being the first tested by the original dataset creator), DenseNet121[31] for representing most of the state of the art solutions and EfficientNet [33] for being a great tool for efficient ConvNet development. Further explanation on the initial configuration will be introduced in 7.1.

Nonetheless, selection of models is just the initial state of the many different design decisions and challenges that creating such a program produces. Since apart from the model selection of proper loss functions, optimizers and many different hyperparameters has to be considered experimentally to aid the Classification Performance of the model into it's best expression of the classification performance metrics.

6.4 Code Breakdown

In this section the main code on which the project's testing was done is explained to understand the different features and flags it has. The code is provided in the github `https://github.com/hugoarsa/TFG` with all the necessary information to be executed.

First of all, the **main.py** is responsible for parsing a variety of flags and arguments that can configure basically everything from directories on which to take and store data to an array of hyperparameters and alternate functions. The main uses are *train*, *evaluate* and *predict*. Model argument selects the desired model between the already presented alternatives with a selection flag between pretrained with ImageNet[13] or not. Next, loss function, optimizer and learning rate scheduler selection is completely customizable too. Finally for training and evaluations it refers to other functions outside the *main.py*. In terms of predictions the main takes a checkpoint models and infers the predicted data getting also GradCAM heat maps[34] for every positive label result with some functions for CAM calculation extracted from a prior implementation from a MIT licensed code *Copyright (c) 2021 Alinstein Jose* in GitHub [65].

The models from *models.py* are all very simple in terms of programming since the predefined model architecture by the *torch* libraries are used with minimal modification to accommodate the multi-label architecture. Overall the modification only involves changing the last fully connected layers or equivalents to another fully connected layer of appropriate size (in our case 14 for all the positive classes listed).

Crucially the training method in *train.py* is a fairly simple and straight forward implementation that uses the described forward pass and backpropagation on an array of batches along a maximum amount of epochs. Loss for backpropagation is calculated with the selected criterion and learning rate is updated accordingly between either batch iterations or epochs according to the different needs of the scheduler. Once the model is updated validation loss is calculated with a validation set and all metrics that can be logged are stored in *history.csv* files for each model. Only if validation is lower than before the best model is stored. There are some considerations in order to give the training process an early stopping capability according to a set patience and epochs can be truncated to a number of iterations for reduced dataset on large testing. *eval.py* contains a similar structure to the validate function but it works with the test set and gives enhanced graphs and results in order to give a more detailed look on the model performance. Train stores model checkpoints at each epoch to restart training if any power outage or unforeseen circumstance happens to resume training after such events with minimal losses.

Finally *utils.py* has an array of original and obtained functions that areproperly cited and from MIT Licensed GitHub repositories. The different functions go from quality of life functions to generate Data Loaders or select between the different configurations of loss functions, optimizers and schedulers to some more particular and detailed functions that run calculations and generate losses.

In terms of the rest of the code we can find: *preprocess.py* to preprocess the original NIH dataset *csv* files, and a jupyter notebook *chestxray_visualization.ipynb* to visualize the

dataset and generate the plots that are seen within the visualization section.

Folder structure contains a *saves* folder for the models histories and plots, *resized_images* for the images of the dataset resized to the desired 256x256 ratio, *labels* folder for the labels csv.

7 Classification Performance Analysis

In Classification Performance Analysis focus will be around evaluating the model's ability to classify and extract information from the data. Because of that the main talking points among the experiments will be metrics like precision, recall, F1 score and AUC-ROC, which are related to the classification ability of the model. That way some insight can be gained on the different model configurations classification potential and see which can provide greater positive impact on the models performance.

7.1 Initial architecture and experiment methodology

Initial architecture will be a stripped down implementation with a very simple model and very simple configuration parameters and methods. This will include a *ResNet18 model* with simple execution and faster convergence times in order to not lose a lot of computational power on the testing and finish them on feasible times. The architecture is similar to one of the final models (ResNet50) and is a simpler version of state of the art models. Simple *SGD* as described in the background as the optimizer with a initial learning rate of *lr*. Finally a loss function of *BinaryCrossEntropyWithLogits* for multi-label classification. This will state the base building block and initial simple model from which experiment through experiment will be shaped into the proposed state of the art implementation for the problem that the thesis focuses on.

Consequently, the methodology for the experiments will be to take the previous base architecture and test for a particular hyper parameter or aspect of the architecture in order to experimentally determine which combination offers the bigger advantage. Crucially, experiments are not set in stone and the initial draft of experiments was enhanced with findings along the way. At first, four experiments where thought but through different shortcomings and problems observed in a single experiments new lines of development where found. Finally, the order is defined by the incremental nature of the methodology.

7.2 Experiment 1: Optimizer

First of all, the first consideration to be taken will be the optimizer selection. Since the optimizer is responsible of updating the different parameters according to certain treatment of the gradients of the loss function generated by backpropagation. In that sense different optimizers might choose different paths for exploring the solution space. For that reason it was thought that it would be the most highly influential element on the training process which makes it a good starting point for the experimentation.

All tests will be taken for 30 epochs to see how the models evolve over time and each epoch will have a fixed 40% size of the dataset through fixed iterations to not waste computation to explore the behaviour of many different configurations.

All optimizers were taken straight from the *torch* libraries without modifications.

7.2.1 Background of Tested Technologies

Main explanations of the optimizers come from the highly synthesized and useful Ruder paper[66].

In terms of plain SGD to clarify the distinction between **SGD Nesterov** and plain **SGD with Momentum** we have to better define the process. Firstly normal SGD performs an update with respect of the previous parameters θ on the opposite direction of the gradients:

$$\theta = \theta - \eta \nabla_\theta L(\theta)$$

The presence of steep curves along some dimensions more than others SGD oscillates due to wanting to go to the steeper dimension but missing the overall local minima. To accommodate for this fact a fraction γ of the last update vector to the current one[66]:

$$v_t = \gamma_{t-1} v + \eta \nabla_\theta L(\theta)$$
$$\theta = \theta - v_t$$

This SGD with momentum is the variant used in our case. Additionally Nesterov variant adds an approximation of the next position of the parameters calculating the gradient of the approximate future positon in order to correct the step accordingly[66]:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta L(\theta - \gamma v_{t-1})$$

Then, **RMSprop** by Hinton [67] is an evolution of previous optimizers like Adagrad that adapted the learning rate to the parameters performing differently sized updates according to frequency. Adagrad did this storing the sum of squares of the gradients up to that point. Nontheless this led to aggressive learning rate diminution. RMSprop stores those gradients in an exponentially decaying average of squared gradients (Variance of the Gradient or Second Moment)[66].

$$v_t = \beta v_{t-1} + (1-\beta)(\nabla_{\theta_t} L(\theta_t))^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} \nabla_{\theta_t} L(\theta_t)$$

Finally, Adaptative Moment Estimation (**Adam**)[63] performs adaptive learning rates too with not only stored mean squared gradients v_t but the mean gradients themselves m_t (First Moment) in a similar fashion:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_{\theta_t} L(\theta_t)$$

Initially the squared gradients and gradients are initialized to 0 and as such the first gradients need to be computed with bias-correction (sing the $\beta 1$ for m_t and $\beta 2$ for v_t. The final expression is similar to the RMSprop one with the added m_t:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Finally **Adamax** (also introduced in the [63]) uses a calculated norm L^p where $p \to \infty$ of gradients as the Second Moment u_t.

$$u_t = \max(\beta_2 v_{t-1}, |\nabla_{\theta_t} L(\theta_t)|)$$

In general consensus over most state of the art papers previously mentioned the two momentum by parameter learning rate annealing that the Adam model offers performs best in terms of fast convergence (and even with it's default betas proposed by the original paper). One tendency mentioned in the Ruder paper [66] points to a simpler SGD solution with learning rate annealing achieves a minimum but it takes some more time to converge. Also SGD is desperately needed for good annealing processes that take it out of local minima when it gets stuck.

7.2.2 Test Results

Table 2: Classification Report Summary for Different Optimizers. Averages are calculated with weighted functions according to support for each class.

Metric	Adam	AdamW	Adamax	SGD	SGD_Nesterov	RMSProp
Test Loss	29.9613	30.1298	30.1492	30.6092	30.2026	33.4058
Avg Accuracy	0.7381	0.7360	0.7279	0.7271	0.7345	0.6607
Mean AUROC	0.8098	0.8049	0.8013	0.7976	0.8043	0.7187
Avg Precision	0.2772	0.2747	0.2703	0.2661	0.2723	0.2263
Avg Recall	0.7137	0.7105	0.7052	0.7011	0.7082	0.6488
Avg F1-Score	0.3797	0.3765	0.3710	0.3661	0.3736	0.3162

From the table of results we can see that clearly over the 30 epochs Adam lowered the loss of the test set to the minimum, closely followed by Adamax and AdamW as they are small permutations of the basic Adam algorithm. SGD is somewhat worse but not radically worse as RMSProp. In future experiments we will keep in mind the possibility of using SGD if Adam seems to falter it's learning. Since SGD reaches a steady convergence with the addition of better learning rate schedulers and might overcome the poor result on this experiment.

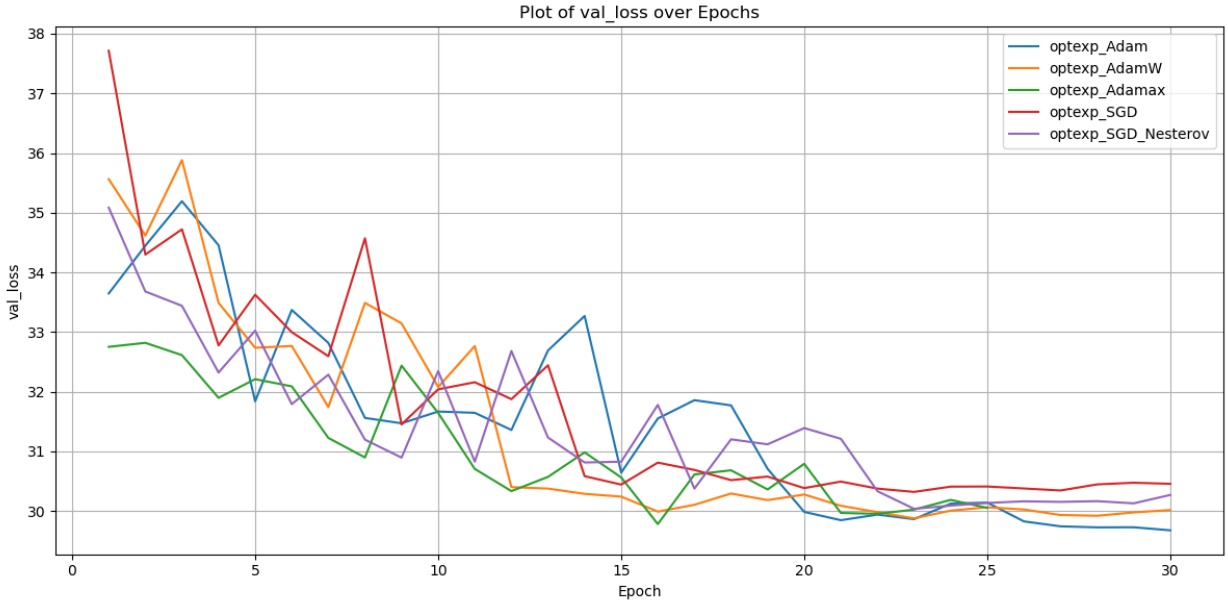


Figure 33: Validation Loss over Epochs with RMSProp removed for better visibility of the remaining losses. The smaller the loss the better. Generated by author.

All other classification metrics seem to behave similarly to the loss in terms of place in the pecking order for the optimizers. This puts a clear advantage for the moment upon the Adam optimizer above it's more similar variants and distant cousins.

Validation Loss over the epochs on figure 33 shows some interesting dynamics. Mostly all optimizers show a serrated pattern that can be partially due to the partial exploration of the epoch through a percentage of iterations executed. Since the random batch order might skew the step that the optimizer takes based on the class balance that results from the seen data for that epoch. SGD starts with very large loss and in the end it plateaus very soon, this can be due to getting stuck in local minima as we previously stated. Related to vanilla SGD, its Nesterov variant achieves better final results with improvements almost to the end. From the other Ada-like optimizers we can see that Adamax performs overall better but fails to keep improving halfway through and Adam keeps improving until the end. AdamW converges faster but ultimately fails to improve significantly once it plateaus.

7.2.3 Conclusion

To sum up, Adam is the best model at the moment since it reaches the best result and doesn't yet show any signs of plateau with validation upgrades until the last epoch. This is very desirable behaviour since it seems like the model can navigate through local minima and find minimum values that are more relevant for the projects solution. The SGD might be of use in later stages of the experiments can lead to better performances and more stable convergence with the right configuration.

7.3 Experiment 2: Simple Batch Size and Learning Rate

Batch Size can have impacts on both computational efficiency, through better fitting data sizes for the internal GPU architecture, and in classification terms, since gradient calculation is done with the whole batch taken into consideration. Because of that finding the best batch for the architecture is highly relevant. The main hurdle about this is that some sizes can fill all the GPU memory available and be impossible to implement. Gradient accumulation is a possible option for accumulating different batches under different a single super-batch in the eyes of classification performance. This approach was deemed unnecessary since extremely large batch sizes (within the capacity of the GPU) did not yield extraordinary values in the first place.

A part from batch size some studies on learning rate where executed in order to experimentally find our best fit. In terms of learning rate it depends a enormously on the selected scheduler, since different schedulers will behave differently on the evolution of the learning rate. In this case since we use at the moment we are using the *ReduceOnPleateau* scheduler from *torch* this learning rate won't necessarily translate to a good fit for the different. As stated in prior studies that use the ReduceOnPlateau method the initial learning rate that is more promising is around 0.001 in papers like CheXNet [10] or 0.0005 in papers like CheXClusion [11]. Our range of tested learning rate contain these two among others to ensure thorough testing.

In order for all tests to process the same amount of images per epoch the number of iterations per epoch to be executed to save futile computing was calculated individually for each batch size to cover the 40% of process data desired.

7.3.1 Test Results

In terms of results it can be appreciated that batch size can have an impact on resulting validation losses if we take a look to table 3. It's not critically dramatic as it seems having a small deviation between tests. Overall 64 Batch size seems to produce better results but the relative upgrade in performance related to the second best in most learning rates does not exceed some thousandths of validation loss. In terms of learning rates we corroborate the 0.0005 value suggested in CheXClusion by some margin.

Table 3: Final Validation Losses for different Batch Sizes and Learning Rates.

Batch Size	1e-4	5e-4	1e-3	1e-2	1e-1
32	0.2625	0.2600	0.2764	0.2986	0.3001
64	0.2594	0.2591	0.2622	0.2977	0.3062
128	0.2681	0.2637	0.2607	0.3009	0.2975
256	0.2637	0.2583	0.2732	0.3092	0.3077
512	0.2664	0.2804	0.0000	0.0000	0.0000

In terms of AUC-ROC score the values follow similar trends to the ones observated in the validation loss table. Since the best learning rate is 0.0005 and the best batch size is 64.

Table 4: Final Validation AUCs for different Batch Sizes and Learning Rates.

Batch Size	1e-4	5e-4	1e-3	1e-2	1e-1
32	0.7761	0.7754	0.7444	0.6298	0.6069
64	0.7786	0.7800	0.7690	0.6351	0.5933
128	0.7733	0.7764	0.7741	0.6195	0.6318
256	0.7691	0.7848	0.7643	0.5960	0.5848
512	0.7652	0.7654	0.0000	0.0000	0.0000

In terms of efficiency it can be highlighted that not much improvement is found on bigger batch sizes, quite the opposite in terms of training times. Since batches of size 32 seem to offer the best times all round with a small dip in performance of about two seconds along the 64 batch size.

Table 5: Epoch Times for different Batch Sizes and Learning Rates.

Batch Size	1e-4	5e-4	1e-3	1e-2	1e-1
32	25.39	24.84	24.73	24.63	24.60
64	26.92	26.48	26.49	26.52	26.43
128	25.46	25.33	25.29	25.25	25.25
256	26.69	26.06	25.93	26.81	26.79
512	77.39	79.43	0.00	0.00	0.00

7.3.2 Conclusion

The conclusion for this test is clear to some extent depending on the desired qualities of the final model. On the one hand, learning rate is very clearly at it's best when it has an initial value of 0.0005 without doubts. On the other hand, batch size seems a little bit divided since 32 offers marginally better times for marginally worse classification and 64 gives the opposite result. As for the final decision regarding batch size it will depend on the final model since the both of them can provide a performance trade-off that will depend on the particular case where the scale tips.

7.4 Experiment 3: Loss Function

Now regarding the different solutions that treat the class imbalance natural to the dataset, we come across the loss function as the main actor on this matter. Since the numerical value of the loss is what determines the quality of a given step within the training process it is vital that the loss accurately depicts the quality of the proposed prediction in a pondered and well thought way.

These loss functions have very different scales in terms of the numerical loss value produces so head to head validation loss graphics or comparisons are not very illustrative. We will need to resort to classification metrics to determine how good each loss function is.

7.4.1 Background of Tested Technologies

At first most of the investigation kept leading to mutli-class problem descriptions like the one presented in the Long Tail CXR dataset [15]. This more conventional approach ensures better performance for overall balancing of the classes but it was no use for out problem description of multi-label. This problem also is based on a more challenging imbalance created from the original NIH dataset with added even more imbalanced labels. In terms of imbalance, we can mainly adapt *re-balancing or augmenting*. In out case, class rebalancing is done both *resampling* the original data to some extent with undersampling and then *re-weighting* through weighted loss functions that adapt the importance of given classes.

Luckily in the realms of weighted loss functions a 2021 paper by Ben-Baruch et al. [20] gives proper attention to multi-label datasets appealing to the significant of pictures in almost all contexts often containing more than a single one posive label. Although positives are usually few against the many negative labels. This can lead to not concentrating enough on positive labels when the imbalance is great and can lead to a reduced accuracy. In that sense class imbalance is not directly tackled as much as this negative-positive balance but it still can yield great results on that remark since by focusing on positive samples more it can focus on underrepresented classes with few positives.

In order to properly understand Asymmetric Loss we must start from the binary cross entropy since multi-label is often reduced to an array of binary classification tasks. We have K labels and one logit output per label z_k whitch is turned into a prediction $p = \delta(z_k)$. Ground truth for class k is y_k. The total loss is the aggregate of all binary losses. A general formulation for binary loss for a particular label L separated by negative and positive loss is:

$$L = -yL_+ - (1-y)L_-$$

Both parts can be calculated with the formulas suggested in the Focal Loss paper by Lin et al. in 2018 [68]:

$$L_+ = (1-p)^{\gamma_+} \log(p)$$

$$L_- = p^{\gamma_-} \log(1-p)$$

With a focusing parameter γ that makes it so the contribution of easy negatives are down-weighted. Finally, in asymmertic loss decoupling is proposed between positive loss and negative loss gamma in order to put more emphasis on positive cases. This gammas have to be found through hyperparameter testing which we will to in this section.

Finally, some shifting of the probability for negative samples is done by applying hard threshold with the defined shifter probability as:

$$p_m = \max(p - m, 0)$$

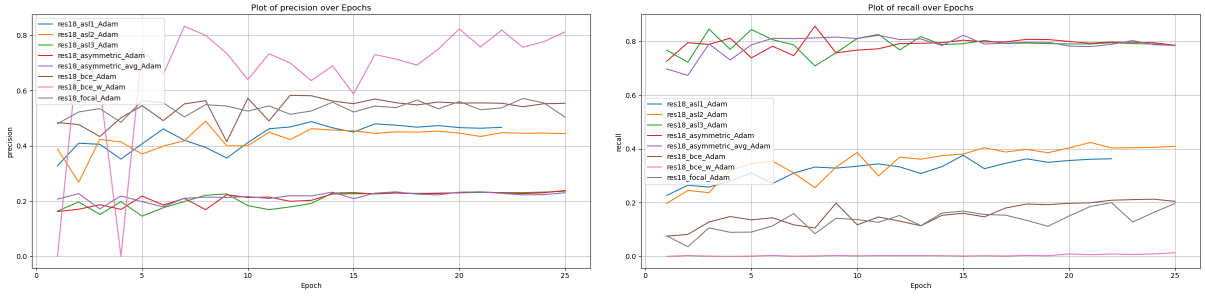
Then we can finally conclude the final loss function for Asymmetric Loss:

$$L_{\text{ASL}} = \begin{cases} L_+ = (1-p)^{\gamma_+} \log(p) \\ L_- = (p_m)^{\gamma_-} \log(1-p_m) \end{cases}$$

On another note, this method has no need to be presented static class weights since it's re-weighting is done at a formulation level on which it works as is like any other mostly parameter-less function. Through smart setting of the gamma values we can obtain binary cross entropy and focal loss with the same loss function which is a bonus.

7.4.2 Test Results

The final tested loss functions are regular BCE, BCE with precalculated weights according to presence, Focal Loss, and several ranges of gammas for the asymmetric loss. The gammas go from small imbalance to big imbalance with some additional hyperparameters clip which controls the previously mentioned probability shifting. We started from the most basic Asymmetric (without any hyperparameters) and build up from there gradually adding imbalance to see the sweet spot.



(a) Precision over Epochs.

(b) Recall over Epochs.

Figure 34: Comparison of Different Loss Functions: Precision and Recall.

The best metric to see how good this loss functions are is f1-score, since when talking about either Precision or Recall there's a certain trade off on which some models that favour precision over recall and vice versa as it can be seen in figure 34.

When looking at F1-scores on figure 35 much clearer conclusions can be drawn and it is clear that asymmetric loss boosts significantly how good both recall and precision metrics are through a higher f1-score. BCE and Focal loss perform quite similarly and BCE with weights seems to have a lot of trouble classifying almost anything. An effort was done in order to try and get it to work but no clear solution was found so it mainly was discarded. All other metrics like AUC-ROC go accordingly to the f1-score giving higher values to ASL. In our case the best ASL was the one that had a $\gamma_- = 2$ and $\gamma_+ = 1$ with $clip = 0.025$.

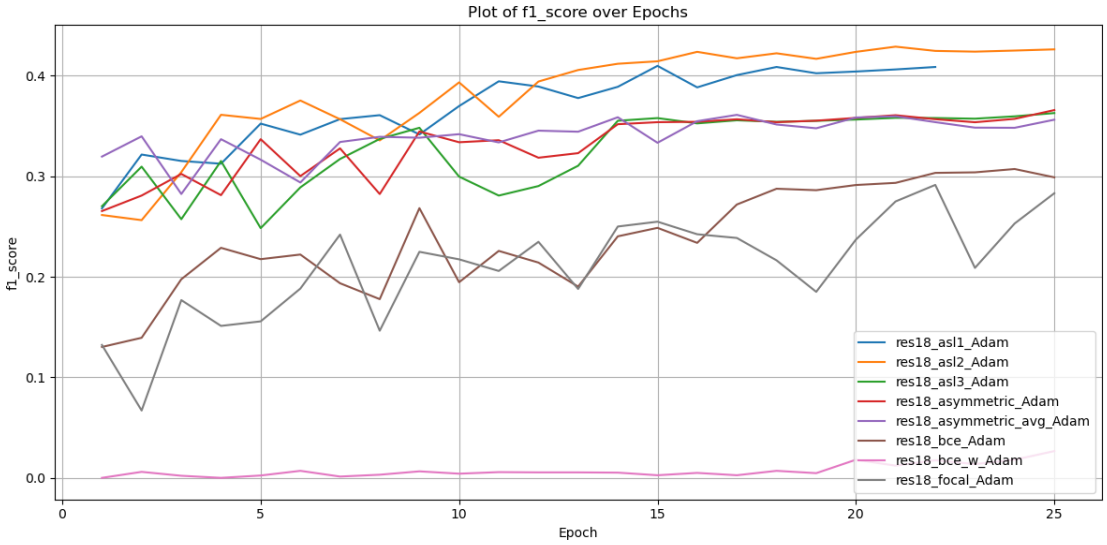


Figure 35: F1-Score over Epochs with different Loss functions.

7.4.3 Conclusion

In terms of loss function Asymmetric loss clearly wins over other types and clearly performs better in it's tuned hyperparameter configuration obtaining better scores in all metrics and ensuring great learning potential. Finally in ROC curves for all classes we can see that in figure 36 Asymmetric loss achieves better classification for Minority classes like Hernia or Consolidation with al labels generally closer to eachother. It's important to focus also that no accuracy on other classes is sacrificed in order for this generally enhanced representation on minority classes that have higher negative-positive imbalance. The harder classes like Infiltration and Pneumonia are not very good at any of them but it's a common trait since Infiltration is a Highly co-ocurrent class that is often mixed with other traits which can shadow it's presence or be confused with it and Pneumonia has a complex nature in frontal RX scans making it hard in general to classify.

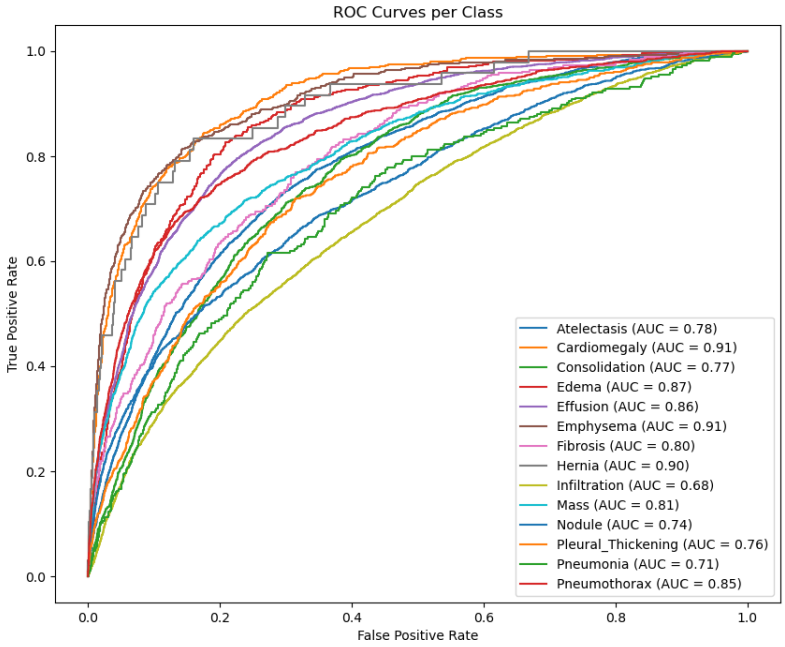


Figure 36: ROC curve for Asymmetric.

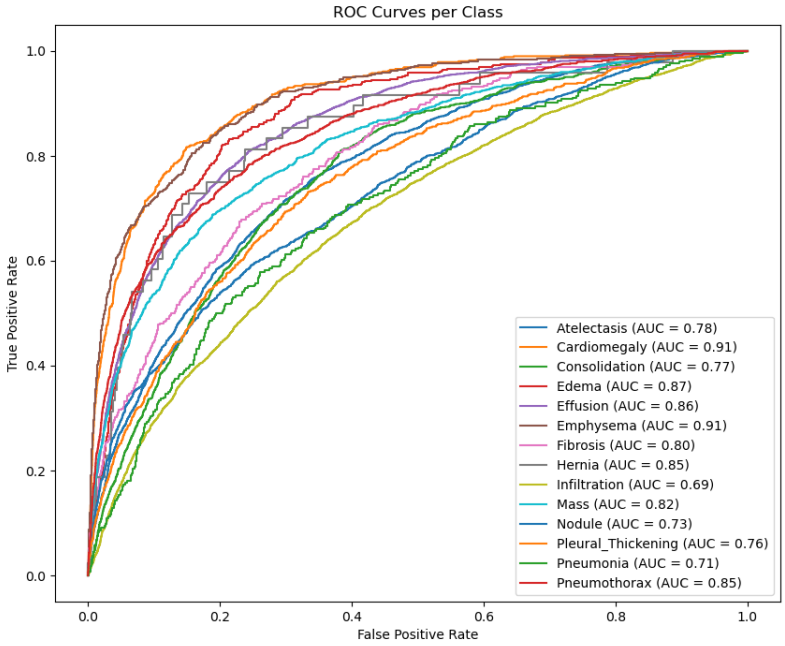


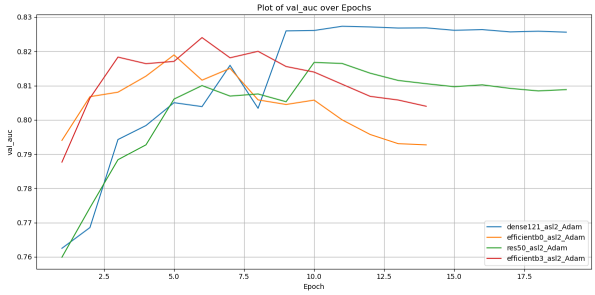
Figure 37: ROC curve for BCE.

7.5 Experiment 4: Models

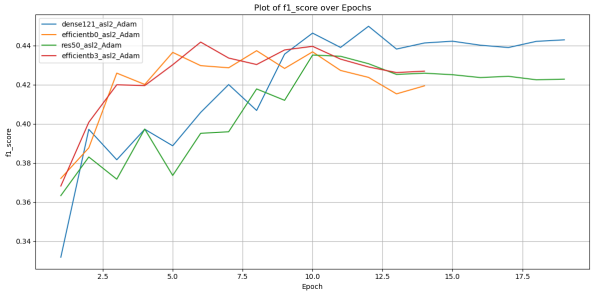
Now that some important decisions have been made in order to try to find the best architecture around a model we try the different models we have in a 60 epoch run with full datasets per epoch to try and get the best results with the previous experiments results. All background is already introduced in the background of the project so we jump straight into the results.

7.5.1 Test Results

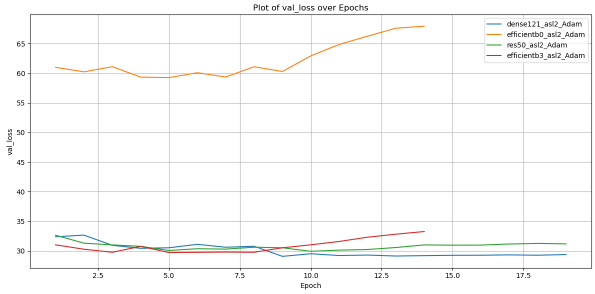
Test results on figure 38 are puzzling to say the least. In general all models triggered early stopping without much opportunity to get good results out of the problem in relation to the previous tests where early stopping didn't seem to be triggering. In general AUC and F1-score are significantly better that in prior tests. This is reasonable since it's training on the full dataset instead of subsets of itself. Nontheless, it seems to reach a very swift convergence within 10-15 epochs and then plateau on it's loss upgrades or even plummet onto worse results. In terms of comparing the models apart from the general early stopping tendency, DenseNet121 seems to be the best one as described in plenty of prior studies[11][10], closely followed by EfficientNets[33] and finally with generally lower results ResNet50. If we take a look at execution times it's interesting to note that EfficientNet really is much better on its B0 size than any of the other and produces similar results to its bigger counterparts. Also EfficientNets reach peak results in a record 5-6 iterations so learning is really fast a part from execution times being low.



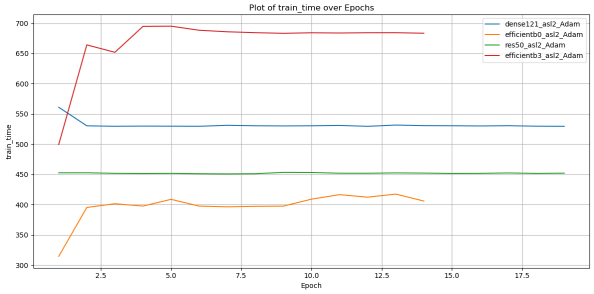
(a) Validation AUROC over Epochs.



(b) F1-Score over Epochs.



(c) Validation Loss over Epochs.



(d) Train Time over Epochs.

Figure 38: Comparison of Models: AUROC, F1-Score, Learning Rate, and Validation Loss.

Finally, we take a look into the table of results for the test set to see how well this training generalize to a test of an unseen dataset in table 6.

Table 6: Classification Report Summary for Different Models. Averages are calculated with weighted functions according to support for each class.

Metric	ResNet50	DenseNet121	EfficientNetB0	EfficientNetB3
Validation Loss	29.9383	29.8123	59.1185	29.8645
Avg Accuracy	0.7461	0.7483	0.7453	0.7479
Mean AUROC	0.8167	0.8233	0.8189	0.8183
Avg Precision	0.2825	0.2882	0.2833	0.2820
Avg Recall	0.7198	0.7257	0.7202	0.7197
Avg F1-Score	0.3864	0.3929	0.3871	0.3859

We can see that the results in the test set are pretty head to head with a small edge on the DenseNet121 model which performs better in all metrics by a small margin. Some very interesting features are that EfficientNetB0 is extraordinarly close to DenseNet121 in most metrics for being such an efficient and computationally inexpensive model. It's also very interesting to highlight that B0 performs mysteriously better than B3 which is very interesting since they are massively different in size and computational impact and yet produce little to no upgrade.

7.5.2 Conclusion

In conclusion the best classifier overall is DenseNet121 like in other studies but it is closely followed by a new contender in the EfficientNetB0. It's similar results in classification and great overall efficiency offer a great alternative to the state of the art model. Particularly, in terms of quick prototyping with other architecture configuration or searching hyperparameters. And in general in terms of easy to train yet immensely powerful models.

A part from model selection the main conclusion is also the alarmingly fast and uneventful convergence on which learning plateaus in a very aggressive way. In that sense, exploration on the learning rate scheduler must be done in order to find better patterns than reduce on plateau to find better minimal values instead of getting stuck in local minima. Future experiments that weren't planned will be added in order to sort this unsatisfactory result and see if we can push the metrics a bit higher in our final proposed model. The proposed plateau at the moment

7.6 Experiment 5: Scheduler

Since our problem at the moment is related to getting stuck in local minima a deep study of learning rate scheduler alternatives was undergone in order to find a better way of preventing our models to lose their learning capabilities so soon in the learning process. This is done in an array of alternatives that all share a common feature: a way of getting out of local valleys through some type of method to bring up the learning rate periodically.

7.6.1 Background of Tested Technologies

The basis of our initial learning rate scheduling is Reduce On Plateau with its *torch* implementation. Its name is fairly descriptive on it's own since it updates the variable when the learning plateaus and a particular watch metric stops improving. In our case improvement is perceived as reducing it's value. This plateau definition is done according to a set patience value that lets the watch metric to not upgrade in a set amount of epochs until the update is made. The update is done multiplying the factor to the previous learning rate. Initially our factor of update to the learning rate is 0.1 with a patience of 3.

Next up, we have the Cyclic Learning Rate Scheduler proposed by Smith [69]. Which, as its name states it performs some type of cycle with the learning rate according to a particular oscillation frequency. This is attractive since it removes the need to find the best learning rate since it fluctuates between a range of learning rates through time. In this paper it is found that this cyclical motion produces overall improved classification capabilities without the extra computation that adaptive learning rates have.

The learning rate policy of Cyclical learning rates is as straight forward as it seems with minimum and maximum boundaries and a stepsize that defines the amount of batch iterations from the minimum to the maximum. The travel between the points can be done with a variety of functions (lineal, parabolic, sinusoidal, etc.) but the paper studies show that it has little impact on the result. The main benefit is that by enlarging the learning rate it has an immediate drawback but in the long term it allows the model to explore better the solution space. Here we can define the triangular oscillating function:

$$\mathrm{lr}(t) = base\_lr + (max\_lr - base\_lr) \times \left(1 - \frac{batch\_iter}{step\_size}\right)$$

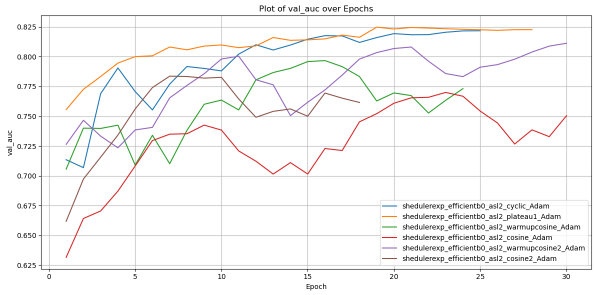
The recommended values by the authors for the hyperparameters are general and should translate to good performance in all scenarios. They propose a minimum learning rate of 0.001 and a maximum of 0.006 and a step size of roughly twice the number of iterations within an epoch. They state in the paper that 3 cycles train almost all the weights and form 4 onwards it achieves better performance[69].

Around the same time another study came up with a similar idea to restart learning rates with the Loshchilov and Hutter 2017 paper [70]. They restart the learning rates with fixed frequencies and then for the reduction part some annealing is done with a cosine function to reduce softly the learning rate to its minimum before being restarted.

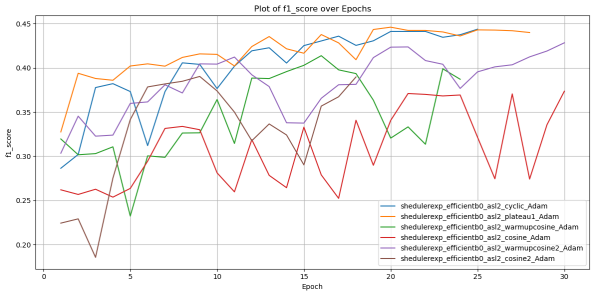
First we used the Cosine Annealing function form *torch* that implements a similar function to the previous cyclic learning rate scheduler with cyclic cosine annealing. But the final implementation used came from a permutation of the original paper idea implemented by katsura 2023 [71] with its warm restarts being gradual. This is implemented with a warmup steps variable that made it so warmups where gradually introduced along a fixed amount of epochs. The function that describes the process is the following:

$$\mathrm{lr}(t) = \begin{cases} \eta_{\min} + (\eta_{\max} - \eta_{\min}) \frac{t}{\text{warmup\_steps}}, & \text{if } t \leq \text{warmup\_steps} \\ \eta_{\min} + 0.5(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t - \text{warmup\_steps}}{T_{\text{cur}}} \pi\right)\right), & \text{otherwise use cosine annealing} \end{cases}$$

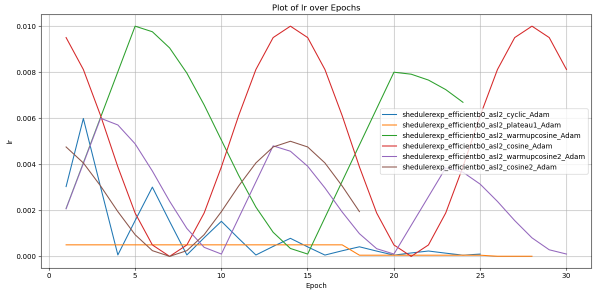
7.6.2 Test Results



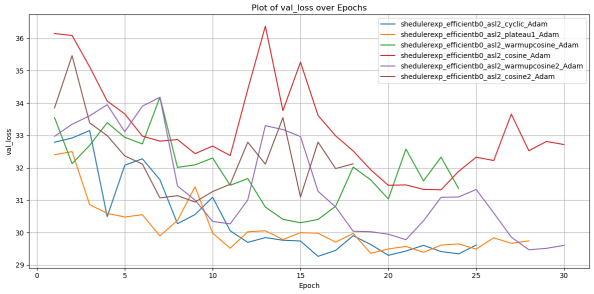
(a) Validation AUROC over Epochs.



(b) F1-Score over Epochs.



(c) Learning Rate over Epochs.



(d) Validation Loss over Epochs.

Figure 39: Comparison of Learning Rate Schedulers: AUROC, F1-Score, Learning Rate, and Validation Loss.

Results in figure 40 show first of all the different learning rate patterns in action with different hyperparameters to test the best model possible. A mysterious result in the validation loss over epochs shows that Reduce On Plateau with it's updated hyperparameters with more patience performs almost as good as the others if not the best overall. One major problem with it nonetheless is that it prematurely stops due to validation loss patience so even though it performs better we can see that it's still plagued by the same problems than before. In terms of other alternatives that are similarly promising we can find the cosine annealing with warm restarts (purple one) which has very interesting validation loss until the very end. This is due to the model finding it's best minimum values at the end of each

cycle. We can see a clear tendency of validation loss getting largerr just to plummet to a new local minima found through the annealing process. This behaviour is shared among all fluctuating methods in general. The decreasing triangular shape of cyclic learning rate scheduler seems to converge beautifully but it's scale might be a bit off because it gets stuck in a similar way than reduce on plateau which is not a promising result.

Table 7: Classification Report Summary for Different Schedulers and Configurations. Averages are calculated with weighted functions according to support for each class.

Metric	cosine_big	cosine	cyclic	plateau	warmupcos_big	warmupcos
Validation Loss	31.5251	31.2762	29.5023	29.4993	30.5304	29.7523
Avg Accuracy	0.7093	0.7133	0.7427	0.7498	0.7272	0.7363
Mean AUROC	0.7732	0.7816	0.8137	0.8212	0.7979	0.8050
Avg Precision	0.2554	0.2582	0.2818	0.2894	0.2718	0.2752
Avg Recall	0.6877	0.6917	0.7179	0.7273	0.7066	0.7107
Avg F1-Score	0.3531	0.3568	0.3852	0.3943	0.3730	0.3773

Now looking at the resulting metrics from the test set we can see that the best results are from plateau even though beforehand it looked like an unlikely contender. Very close in validation loss is cyclic learning followed by warmup cosine annealing.

7.6.3 Conclusion

In conclusion, seems like more patient plateau helps better find a better minimal value but as it's seen in figure 40 it has some problems with getting stuck in local minima still, since once it reaches a deep enough value it fails to improve significantly or does so in a very slow manner. Otherwise, its closest competitors in cyclic and warmup cosine annealing have worse results in general for the tested time but don't show any signs of stalling learning with is highly interesting bearing in mind that plateau early stopped. This might be because they haven't completed enough cycles to reach full potential but there's no sign to think that they are close to not finding improvement. We will keep close attention to these methods in case plateau starts to work poorly again. Also a major oversight of the experiment that was corrected on further experiments is the use of Adam which even though it is affected by the global learning rate it uses its own learning rates for each parameter so impact on the annealing methods might be dampened by this.

7.7 Experiment 6: Image Size

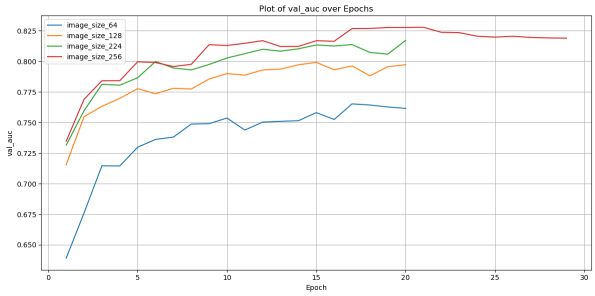
In this section, an assessment how Image Size impacts the whole learning process will be made in order to test general feature loss due to resizing. Since most models have adaptive layers in their finishing convolutional steps in order for any size of image to fit through the particular classifier section multiple image sizes can be fitted effortlessly through the same CNN. Tested sizes go up to 256 since bigger sizes aren't feasible because of quadratic evolution of execution time according to size without a significant increase in performance. Furthermore, state of the art methods like CheXNet [10] downsample to 224x244.

7.7.1 Test Results

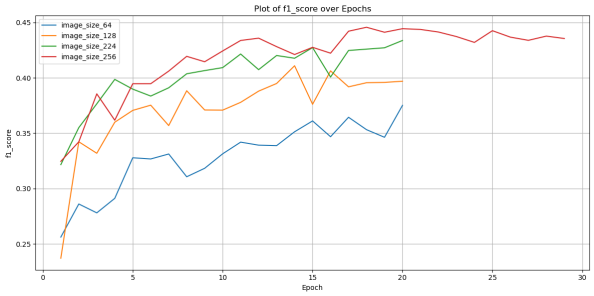
As we can see in the results generally bigger sizes tend to have better results. The step from 64 to 128 is very significant across all metrics but then from 128 to 256 step is significantly smaller. Specially in terms of validation the difference between the top three classes is diffuse until the very end where image size 256 seems to find some extra performance where no other did finding the lowest validation score.

7.7.2 Conclusion

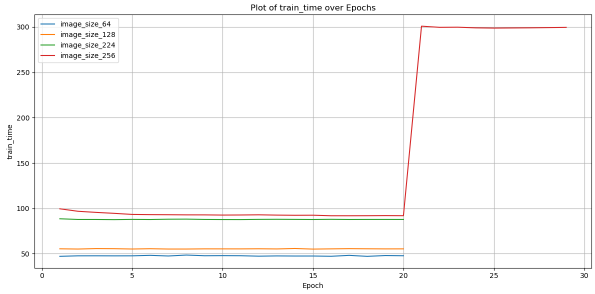
In conclusion the best result in general comes by the hand of the 256x256. Plus, the closeness between sizes in terms of validation loss helps us further affirm that the immense extra computation of doubling image size does not introduce any major upgrades.



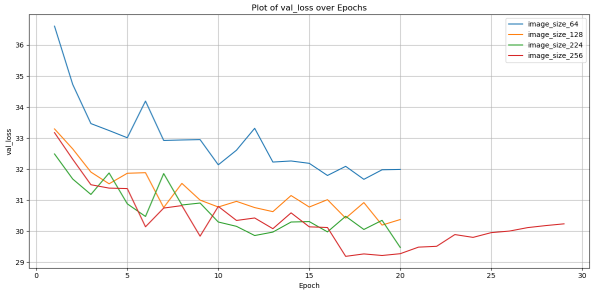
(a) Validation AUROC over Epochs.



(b) F1-Score over Epochs.



(c) Train time over Epochs.



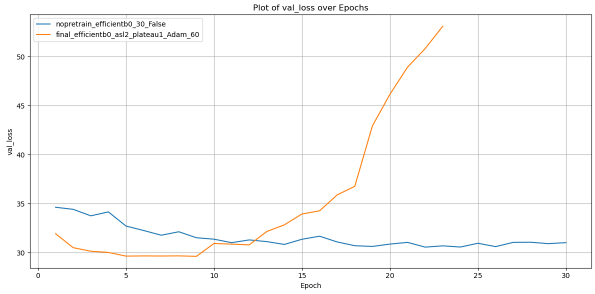
(d) Validation Loss over Epochs.

Figure 40: Comparison of Image Size: AUROC, F1-Score, Execution Time, and Validation Loss.

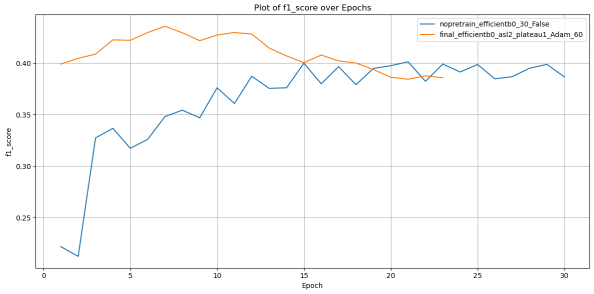
7.8 Experiment 7: Transfer Learning

Finally a short assessment was made in terms of transfer learning impact just to prove that no matter how distant the original training is deep learning models have a great capacity to reuse knowledge obtained from previous training. In our case models are trained by the ImageNet dataset [13] which has 3.2 million images and 5247 classes annotated with the images. The technique is at the heart of Machine Learning and Deep Learning since 1976 as pointed out by a 2019 history review paper by Bozinovski [72]. Nowadays, transfer learning is really straight forward and specially with *torch* you can just load the model with pretrained weights and go as is. All previous experimentation has been done with transfer learning by default.

7.8.1 Test Results



(a) Validation Loss for no pretrained dataset.



(b) F1 score for no pretrained dataset.

Figure 41: Comparison of Transfer Learning: Validation Loss, F1-Score.

Metric	Pretrained	No Pretrained
Validation Loss	29.8346	30.7968
Avg Accuracy	0.7392	0.7135
Mean AUROC	0.8090	0.7856
Avg Precision	0.2806	0.2603
Avg Recall	0.7174	0.6924
Avg F1-Score	0.3834	0.3583

Table 8: Comparison of metrics related to pre trianing.

The results on figure 41 comparing head to head no pretrained models with a pretrained alternative point out that when starting from the model that is initialized to zero the initial loss is worse from the start and has a slow but steady learning process until it plateaus at the end where no improvement is found. This particular case for the pretrained method seems to suffer from poor convergence producing bad final results but even though in just 5 epochs it produced a clearly better result. Also in table 8 we can see that all metrics are strictly better by a significant margin on the test.

7.8.2 Conclusion

In conclusion we can see that ImageNet pretraining a part from being a state of the art standard generalizes well with our particular dataset. Even though we can see some convergence problems this have been proved to be solved on previous experiments on learning rate.

7.9 Experiment 8: Models Final Training

In terms of the final testing for the models all the previous conclusions of testing are put to test on a general 60 epoch test to see how the results of previous tests work with full training sets and long executions. In that sense training can be larger than 60 epochs if needed but most models converge fast enough so 60 was the chosen limit with right to revision at any time. In that sense some results where somewhat mixed with the better performing option showing some potential problems a part from the main metrics. This includes of course the learning rate schedulers which even though fluctuating schedulers where sub-optimal in terms of performance metrics showed fewer signs of plateauing at the end of the 30 epoch test. For that reason, the best possible options for *scheduler* and *optimizer* combos will be explored in order to find the best model overall within our experimentation. Among those mainly we will be testing **Plateau with Adam**, **Cyclic/Warmup Cosine with Adam** and finally **Cyclic/Warmup Cosine with SGD** to test if the better convergence mentioned in the papers is met. Also in terms of models, **DenseNet121** and **EfficientNet** almost tied in performance terms so they both should be tested to further explore the similar performance. Finally in terms of loss functions there is no doubt that Asymmettric performs better in general since the conclusions where fairly clear.

7.9.1 Test Results

At first glance looking at table 9 we san observe that DenseNet121 outperformed EfficientNetB0 when using the plateau scheduler but failed to take profit of the annealing of the Warmup Cosine or the fluctuating nature of the Cyclic with the better convergence of SGD. Because of that, the highest result overalls comes by hand of the EfficientNet with Cyclic scheduler and SGD with an impressive 0.8225 AUC-ROC average and a close to 75% Accuracy in general. Precision is not great but recall is very high ensuring the least amount of true positives is missed. Since it is a medical diagnostic application that is supposed to be use to support the professional radiologists it is more beneficial to have a model more sensitive to positive in order to bring attention to a potential area through the GradCAM heatmap.

Metric	**DenseNet121**			**EfficientNetB0**		
	P/A	**WC/A**	**C/SGD**	**P/A**	**WC/A**	**C/SGD**
V. Loss	29.7247	31.6955	32.7914	29.8346	29.5733	29.4822
Avg Accuracy	0.7422	0.7087	0.6864	0.7392	0.7504	0.7459
Mean AUROC	0.8150	0.7724	0.7504	0.8090	0.8166	0.8225
Avg Precision	0.2800	0.2516	0.2414	0.2806	0.2866	0.2851
Avg Recall	0.7168	0.6825	0.6689	0.7174	0.7241	0.7227
Avg F1-Score	0.3832	0.3481	0.3349	0.3834	0.3910	0.3894

Table 9: Comparison of metrics of test evaluation across different models and optimization strategies.

In terms of learning through the validation loss in figure 42, we can see that we are still troubled by the quick convergence problem on which most models seem to find the best

result soon and then lose track of the minimum values. Our best performing method in brown can be seen converging quickly and steadily without many fluctuations with a very early stop at 22 epochs. This is quite interesting since we are using cyclic methods that are supposed to have better results. They do have better results in relation to the other ones but they still suffer from a relatively short learning process. Also we can observe that plateau seems to have a tendency to miss the local minima and launch off to higher values with makes it unlikely. This tendency was found in previous experiments but is further reinforced in this one. Finally, overall DenseNet121 doesn't seem to be the best option specially in regards to using cyclic schedulers which don't seem to find the same benefits than with the EfficientNetB0 architecture. This could have something to do with the dense nature of the network making it more susceptible to those large changes in learning rate. It would be worth it to look further into it in future works. Also we can see that warmup cosine and plateau schedulers reach lower validation losses but get worse test losses which signals some overfitting happening potentially.

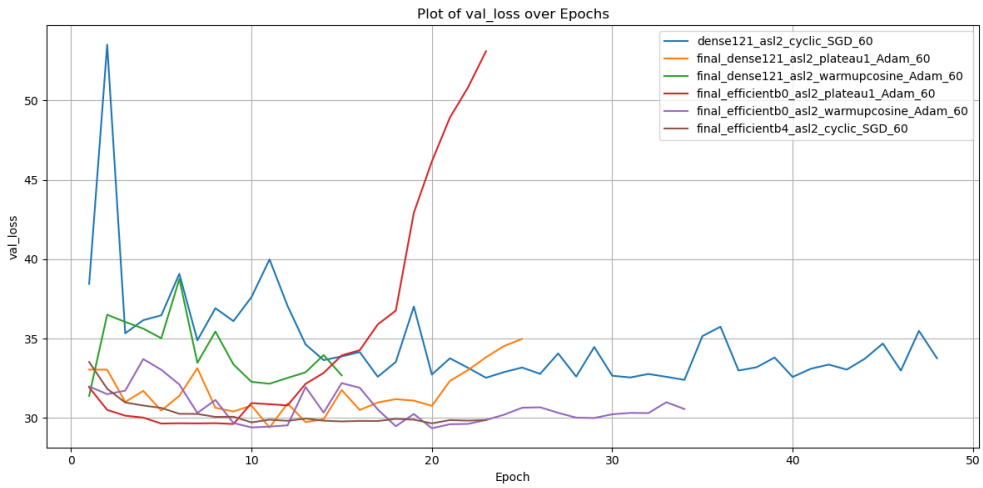


Figure 42: Validation Loss for final models.

Now we will look at the classes that yield better results invidually with the table 10. First of all, can see that we succeeded in making small classes like Cardiomegaly, Edema, Ephysema, Fibrosis and Hernia have decent Accuracies and AUC through Asymmetric Loss Functions. In terms of our less successful classes, we can point at Infiltration and Pneumonia which as presented in the dataset explanation are the harder ones in terms of confusion with other pathologies and as such they don't behave very well in the classification task. Even though they have plenty of positive examples for the model to learn from they stick out as the harder ones to classify in our solution. We can appreciate that the high recall lower precision is present in all classes within the labels so that's good. Upon inference some smart thresholding could be used in order to define better thresholds to find the perfect desired balance between precision and recall but at the moment high recall seems like a fine compromise for the application.

Class	Precision	Recall	F1-Score	Accuracy	AUC	Support
Atelectasis	0.3213	0.7281	0.4458	0.7286	0.8019	2207
Cardiomegaly	0.1799	0.8333	0.2959	0.8335	0.9111	618
Consolidation	0.1448	0.7087	0.2404	0.7107	0.7719	951
Edema	0.1250	0.8194	0.2168	0.8175	0.8814	454
Effusion	0.4225	0.7771	0.5474	0.7771	0.8596	2553
Emphysema	0.1572	0.8324	0.2644	0.8330	0.9190	531
Fibrosis	0.0635	0.7253	0.1168	0.7288	0.8181	364
Hernia	0.0124	0.7917	0.0243	0.7930	0.9185	48
Infiltration	0.4115	0.6444	0.5023	0.6446	0.7005	4097
Mass	0.1805	0.7361	0.2900	0.7365	0.8144	1076
Nodule	0.1748	0.6898	0.2789	0.6899	0.7629	1280
Pleural Thickening	0.1010	0.7045	0.1766	0.7055	0.7840	660
Pneumonia	0.0368	0.6497	0.0696	0.6532	0.7076	294
Pneumothorax	0.2197	0.7902	0.3438	0.7910	0.8637	1020

Table 10: Classification Report per Class of Best Model.

Finally we can take a look at the ROC-Curves of all classes of our final model in figure 43. In the different ROC Curves a simple visual representation of the general ROC performance can be seen reinforcing previously stated groups of better performing classes. We achieved to represent in the same quality some minority classes in the same quality of other more represented classes but failed to improve in some medium tier classes.

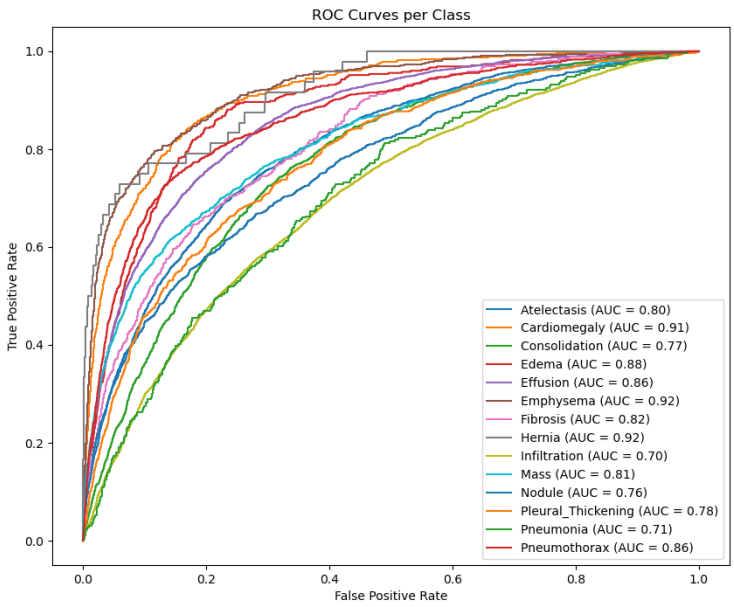
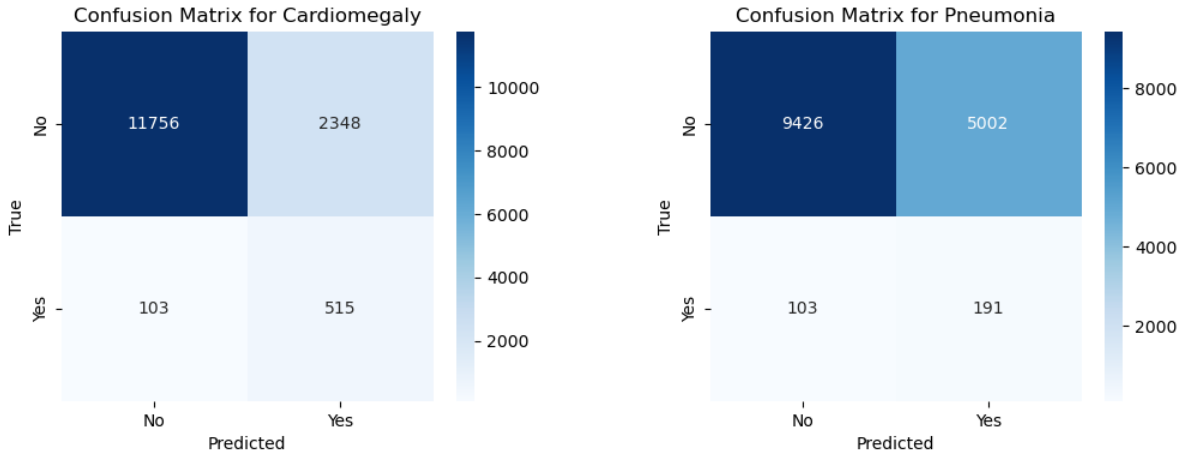


Figure 43: Final Model Roc Curves.

Two confusion matrices of the best performing class "Cardiomegaly" in figure 44a and the worst performing class "Pneumonia" in figure 44b to see how they classify the different labels and their true value. We can appreciate better the precision and recall metrics visually seeing that even better performing variables like cardiomegaly have more than 2000 false positives. The main point is that the amount of false negatives is as low as possible missing just shy of 100 in both labels. Pneumonia is a very alarming case with 5000 false positives but it's due to it's uncertain nature with only frontal RX. Pneumonia needs to be evaluated with symptoms and other complementary tests so it is natural to have poor performance.



(a) Confusion Matrix for Cardiomegaly.

(b) Confusion Matrix for Pneumonia.

Figure 44: Confusion matrices for the best and worst performing labels in multi-label problem.

7.9.2 Conclusion

In this particular experiment, we can see that all the previous decisions have given us an informated opinion on the different elements that can be shifted within the pytorch architecture to maximize our models performance. In our case, decent balancing efforts have met results in the shape of a well balanced accuracy and metrics over all variables. Our final model of EfficientNetB0 with Cyclic learning rate scheduler, SGD, Asymmetric Loss Function and adequate hyperparameters performs significantly better than all other models and converges in a stunningly low amount of epochs. This might show promise for future work to try and find maybe a deeper minimal value with smarter annealing methods or other optimization strategies but within the scope of the project this is the best that we have been able to achieve with our model. Our final model is therefore in the mix with other previous state of the art solutions like CheXNet[10] in terms of classification performance.

7.10 Experiment 9: GradCAM Showcase

Finally some remarks on the GradCAM [34] generated upon inference some different results are going to be evaluated for the quality of their localization ability. As we can see from figure 45 most of the visual explanation produced are of great use to determine the particular element that raised suspicion for the label. All these are correctly placed labels on a particular case.

First of all, the Mass and Nodule labels are greatly localized with the importance focusing on the small an rounded figures on the respective lung. This is to be expected since nodules and masses have a defined visual appearance and the resulting bodies are of high contrast with the dark background. In terms of Cardiomegaly we can see another very successful visualization which is simple in this case due to the fixed nature of the pathology. Since it always appears in the center of the image it is easy for the model to just react to an enlarged central space and thus correctly localize it. In terms of more generalized conditions like Fibrosis which generates general plugging in the bronchioles as previously sated in the illness definition. We can see that this produces a generalized pattern of opacity. Effusion is normally localized in the lower parts of the lung in upright RX scans as we find it in the image. Finally pleural thickening is more subtitle and can happen all over the pleura space (which covers the whole lung) but it usually is seen along the borders of the lung as the localization tries to highlight.

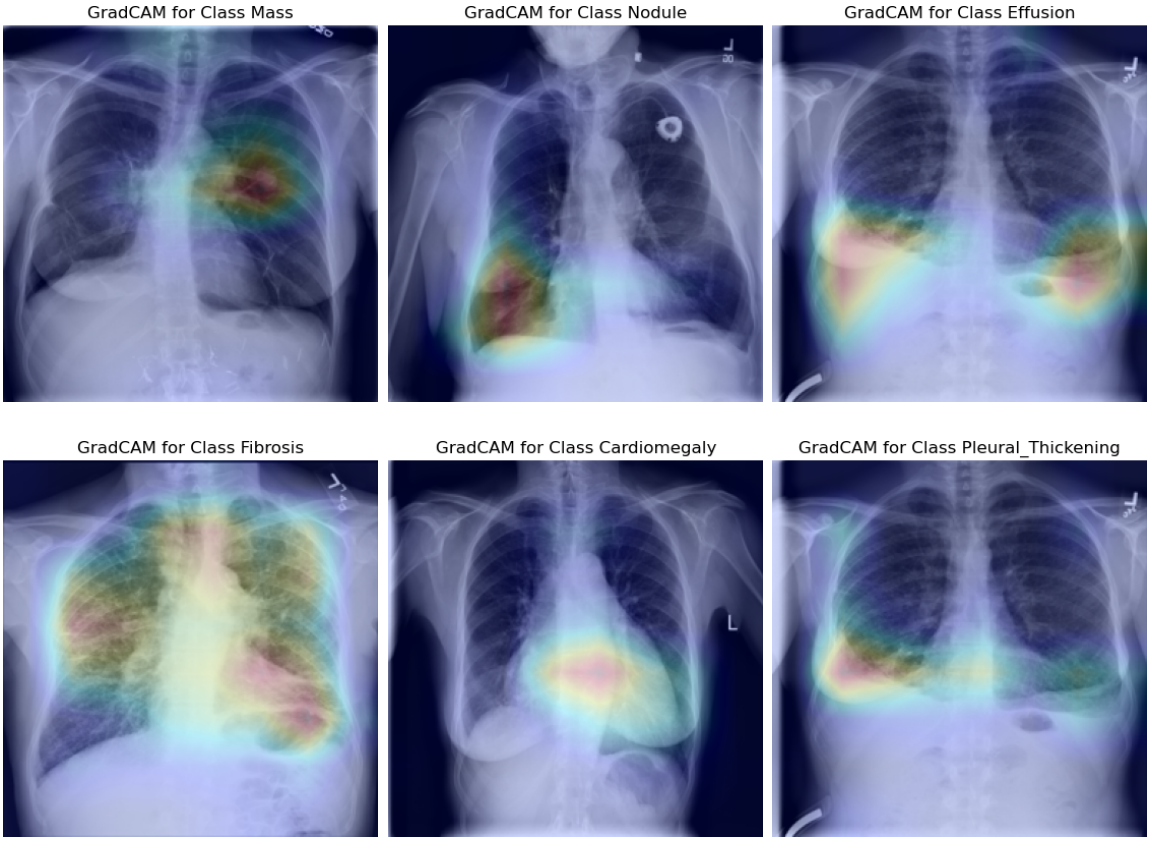


Figure 45: Sample of visual explanation for six different labels with moderate success.

8 Computational Efficiency Analysis

In Computational Efficiency Analysis performance is measured in terms of the computational resources used like processing time, memory usage and energy consumed by the model's training in order to asses also the model's impact on the environment. In this section experiments will try to maintain as much classification performance from the previous experiments as possible trying to get a configuration that finds a good compromise between performance an efficiency. It might be possible that little upgrade is available to be extracted since kernel modification and optimization strategies are too complex and out of scope. Nevertheless, properly quantifying the impact certain parts of the algorithms and determining competent efficiency metrics for our solutions will be of value for future work or other colleagues that have expertise in the field.

All traces are generated with Nvidia Nsight Systems software. In order to get all the different values it is very important to properly call the different functions and prepare the code to be profiled. We use nvtx from the NVIDIA Tools Extension Library to create different labels for the different parts of the code so the trace can be understandable at a high level. This is due to the granularity of the study we will carry out. Since deep cudnn function study is not in scope due to the complexity of the torch library underlying structures. Because of that, our efforts will be on trying to find the best configuration of flags or architecture decisions that produce better efficiencies.

Traces will contain a single iteration of the inner training loop for a single batch. In order for the batch to be properly representative some cool down iterations are left out of the profiling in order to minimize the impact of page and cache misses at the beginning of execution. The traces exposed on the following sections are split into two distinct figures. First, a python activity overview figure will introduce all the different sections of code that are executed each time. Here we can also find CUDA Hardware Activity (CUDA HW) and the activity of different threads for pytorch (only the activity of the main two are viewed. CUDA API calls are also presented in this trace. Secondly, a graph containing different metrics related to Hardware usage and efficiency are presented. Mainly, CPU usage, GPU Clock frequency, GPU activity. Additionally some remarks on Streaming Multiprocessor (SM) activity, instructions and warp occupancy are introduced to see how this hardware sub units are managed. Finally, DRAM and PCIe Bandwidth.

8.1 Experiment 1: Model Architecture Performance Analysis

Now that we know how the different models relate to each other in terms of classification performance a deeper look on how do they work with the GPU hardware will be taken in order to analyze what use do they make of it. This will shed some light into how models administrate their workloads and how that affects the hardware. Mainly we will be looking out for potential synchronization that is not necessary or void spaces where nothing is happening.

8.1.1 Test Results

The main takeaway from the traces over all three models is that it seems like when Data Loading operations are taking place the GPU is not busy with any work and could take extra load. This can be seen for example in figure 47, where at the left side a white space can be seen where no useful calculations are being done. Also we can see at that same figure that fore some brief amount of time before the GPU stops working the load on CPU is low so some time could be exploited if the data loading happen within that time. To further reinforce this concept we can take a look at figure 46 to see that indeed for about 30% of the iteration the CPU is not doing any particular work apart from finishing some calculations untill it runs out of things to do and lowers it's usage until the cudaStramSynchronize is done. This is due to the GPU finishing the different cudnn operations that have been casted by torch and the python code just remaining idle until the data comes back (since in order to do the next batch the weights need to be updated to do the forward pass. Execution time on the DenseNet121 iteration is 350-400ms. We can also see in the tiny red square in both figures at the beginning the data transfer through PCIe is very fast and small, since not a lot of data transfer has to happen from Host to Device.

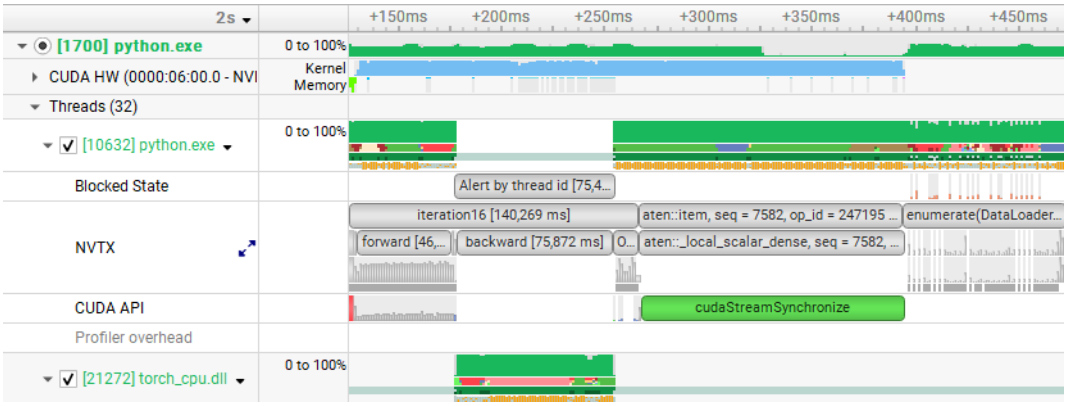


Figure 46: DenseNet121 Trace with python calls and nvtx labels for different phases.

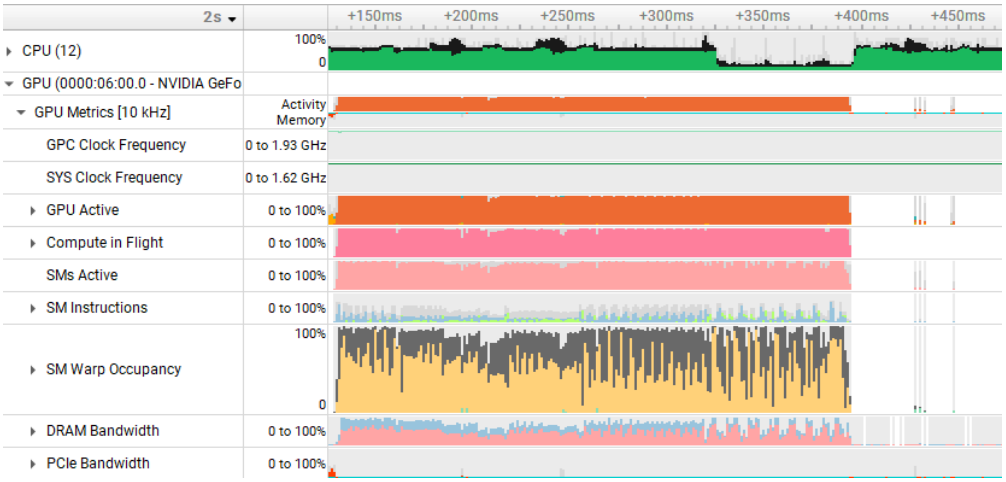


Figure 47: DenseNett121 Trace with GPU utilisation metrics.

Now that we can take a look at the ResNet50 traces in figure 48 we can't see much difference a part from a slightly different distribution and execution time (roughly 300ms). It can be highlighted that the synchronize is way bigger for this one even though it takes less time for the iteration to be completed. This might be because for ResNet50 the python operations that need to be executed are simpler in nature to the DenseNet121 and python just end up waiting more because of it. It can be appreciated too that SM Warp Occupancy in 49 is not as optimized as for the first one since we can see more of the gray space with is allocated unused space at the warps. This might be because of the ResNet50 network being deep and narrow doesn't occupy a full warp with each of it's layers. This in turn leads to worse performance since layer calculation space is being wasted.

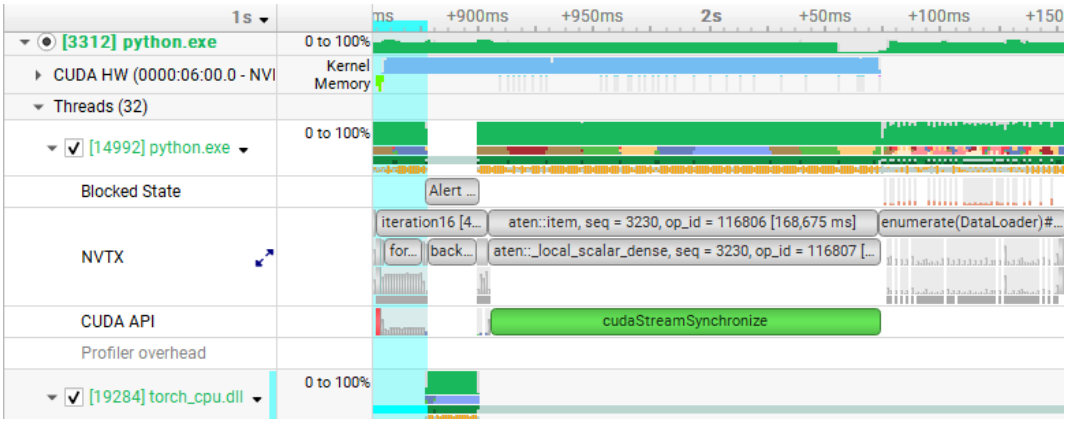


Figure 48: ResNet50 Trace with python calls and nvtx labels for different phases.

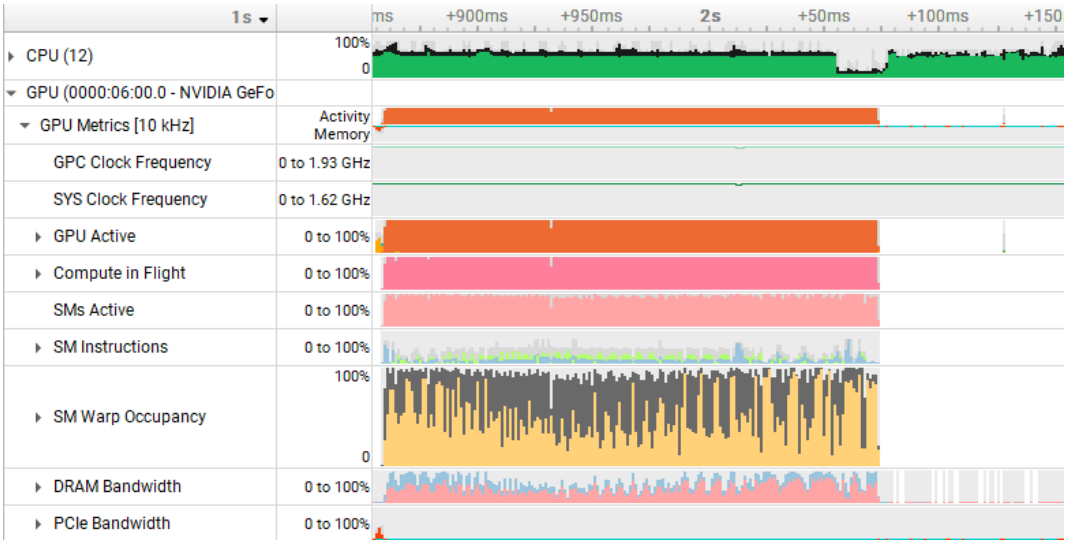


Figure 49: ResNet50 Trace with GPU utilisation metrics.

Finally with EfficientNetB0 at figures 50 and 51 we can see the best distribution of all three with correlates with the more efficient times we have seen. SM Warp Occupation is splendid due to the dimensions of the neural network being proportional to eachother. That way different calculations needed to be done are wide enough to fill most of the warp and use it's computational power more efficiently. A part from that we can see one of the lowest stream synchronize calls for the batch size and a very small execution time of 210ms. With the trace we can clearly see why the design decisions of EfficientNet where taken and how good it's performance is related to the other models.

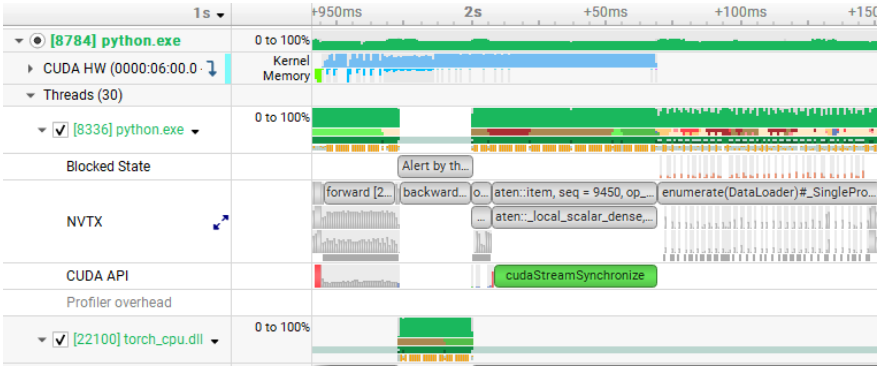


Figure 50: EfficientNetB0 Trace with python calls and nvtx labels for different phases.

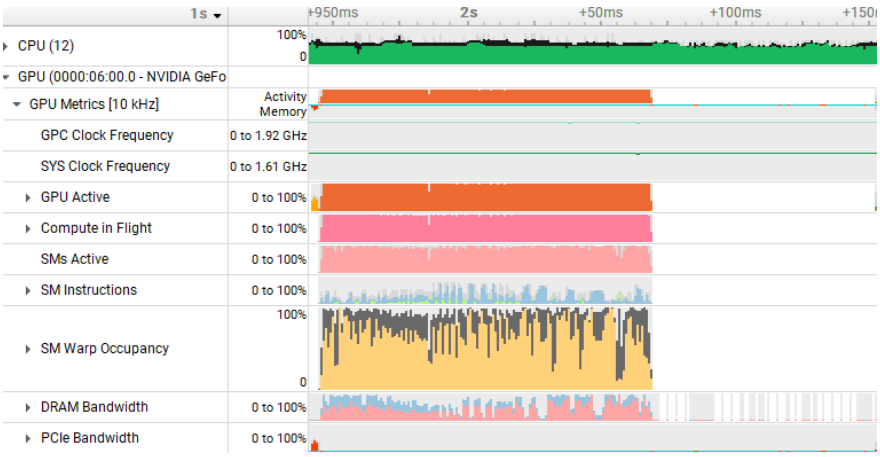


Figure 51: EfficientNetB0 Trace with GPU utilisation metrics.

8.1.2 Conclusion

To conclude this section, we can state now with more confidence why EfficientNet has such enhanced efficiency overall with the better GPU utilization through better Warp Occupancy. Also we have detected some potential issues with synchronizing times leading to diminished use of CPU while the GPU is finishing work and later GPU fully stops working while the CPU gets the batches of future iterations. This are clear sections where the parallelization could be done better.

8.2 Experiment 2: Batch Loader Modifications

Picking up right where we left off in the last experiments now we will try to find performance through the Data Load process since it is in that part of the python code that the GPU stop working entirely and wastes about 75ms waiting for the process to finish.

In order to fix this situation we go to the official torch documentation to look for any possible alternatives to our current Data Load management. We can see that we can assign a particular number of workers through the parameter *num_workers* which creates sub-processes that are used for data loading in order to take the computational load of loading the datasets out of the main process. This is critical in our case since we can get those processes to work while we wait for the GPU to finish in order to skip that CPU overhead that kills the performance on the GPU side. Another related variable is the *prefetch_factor* which states the number of batches to be loaded in advance by each worker. We will keep this value to it's default of 2.

8.2.1 Test Results

As we can see in figure 52 now we have no overhead by the CPU Data Load instruction and we just spend the necessary time of copying memory from host to device. Now that we have workers that copy time is significantly larger than before but it's still significantly less time than before with only 30ms of overhead between iterations. This can be seen at the GPU occupancy and activity measures on figure 53 since now the waiting idle time is significantly smaller. A couple of values for workers where tested and 2 workers where the best configuration possible.

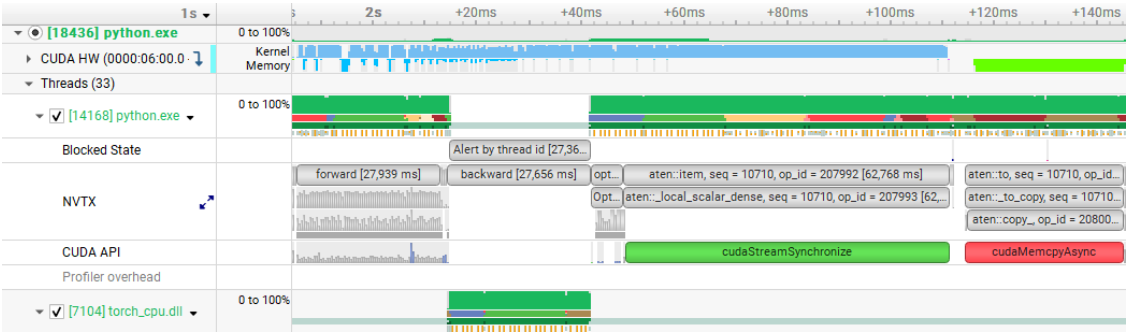


Figure 52: EfficientNetB0 Trace with Threaded Batch Loading with python calls and nvtx labels for different phases.

8.2.2 Conclusion

This experiment was a huge sucess since even though new overheads have to be introduced for multiple workers the results ends up shortening the iteration time significantly producing a noticeable efficiency increase even at high level with execution times for epochs getting about a 20% time save overall.

Figure 53: EfficientNetB0 Trace with Threaded Batch Loading with GPU utilisation metrics.

8.3 Experiment 3: Automatic Mixed precision

Finally, we have to try to use mixed precision in order to try to take profit from the enhanced floating point operations introduced alongside Tensor Cores in 2017. As described in pytorch blog [73] mixed-precision training can use a combination of FP32 and FP16 at certain points of training achieving similar accuracy to the single precision. This is promised to produce shorter training times and lower memory requirements to fit bigger batch sizes in limited GPUs. This second feature in our case is not necessary since we are using the optimal batch size for our model and it fits within our GPU VRAM and the models also fit so we are looking for a better training time. The feature performs automatic conversion between operations to try an improve performance at certain points.

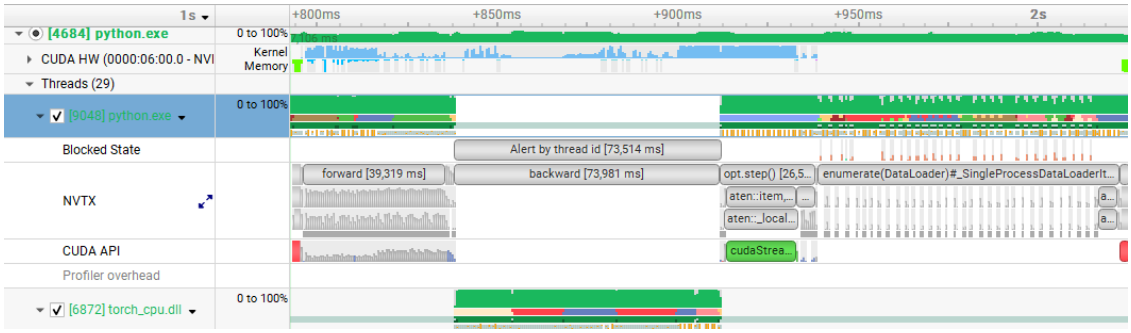


Figure 54: EfficientNetB0 Trace for AMP with python calls and nvtx labels for different phases.

8.3.1 Test Results

We can see that in efficient net the resulting time for a single batch is around 230ms where the original time of figure 50 when now that same iteration spends 250ms to complete as seen in figure 54. Particularly the backwards part of backpropagation is significantly slower than before. This is reinforced in figure 55 where we can see that Warp Occupancy and general GPU activity is at it's lowest at that window of time. It seems that FP16 is being

used since it uses less of the GPU overall and spends similar time. But the main problem is that free computational power is not being used to shorten execution times.

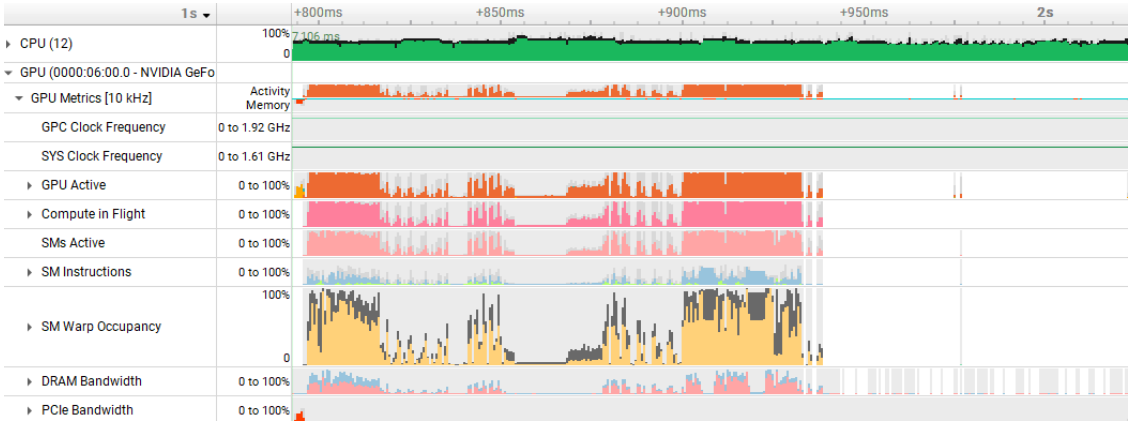


Figure 55: EfficientNetB0 Trace for AMP with GPU utilisation metrics.

8.3.2 Conclusion

Finally, we can see that no upgrade was found within the configuration of the AMP half precision format. This would need deeper profiling by a Compuer Architecture specialist to find potential inconveniences with the pytorch halved precision. In that sense the final solution won't contain the AMP mixed precision format since it didn't bring any performance speedup.

9 Project Management

9.1 Time Management

In this section, a time management study will be carried out in order to ensure the correct timeline for the project to be executed successfully. In order for that to happen, some time constraints will be taken into account alongside different potential risks so that a realistic plan is found.

Since this Bachelor's Thesis we can get a rough estimate of the hours of dedication and work that this project's so that we know how much weekly dedication will be necessary. The project should take 450 hours of work from start to finish (since at Barcelona School of Informatics Bachelor's Thesis are equivalent to 18 ECTS and normally each ECTS corresponds to 25 hours).

The project started at February 19th 2024 and will aim to finish at June 2nd 2024. This will give an extent of 15 weeks from start to finish and will give a comfortable 23 days of cover from project finish to the first available defense date. This time estimation will allow for any problems to be managed before that date in order to be able to make it in time. This much time is necessary bearing in mind the potential risks that will be talked into later in the risks section.

Now that the amount of hours and weeks is calculated a project weekly dedication plan can be carried out. This is specially critical bearing in mind that the author has a very full week with work and classes to fit in Thesis work. If all days of the 15 weeks are taken an average of 4 hours per day. The issue being that in working days the average workload is already 6-9 hours and it might not be realistic to assume that degree of dedication daily. Weekends will be factored in in order to fill the missing hours.

In the end health issues interposed with the completion of the project for the initial date so the autum turn of reading was chosen. Resulting in the final 21th of October date. Time management was similar to the original plan just shifted because of a delayed start.

9.2 Description of tasks

In this section all the different tasks that compose the whole project grouped by general areas of work that have different properties or roles that are assigned that particular task.

9.2.1 Resources Used

Before introducing the tasks that this project will need to be completed, a brief introduction to all the resources that will be needed to carry out the tasks will be explored to better understand the latter tasks.

Human Resources (People)

In terms of Human Resources, even though there's a single author roles will be listed to realistically depict the different functions that the author will take.

- *Project Manager* - A project manager does the actual planning that needs to happen in order to assign all resources and design tasks. This role is responsible of leading the project and putting all the management needed for the project to happen and succeed.
- *AI Solutions Architect* - This role has to carry the design and theoretical weight of the project being responsible to apply prior knowledge and investigate new solutions in order to propose new solutions, evaluate critically the obtained results and produce accurate scientific documentation about the project.
- *Programmer* - This role has to implement and program the described programs that it's superiors have described and be able to manage and prepare the execution environment on the HPC machine.
- *Tester/Data Analyst* - This role needs to be an expert in AI testing solutions in order to be able to design testing environments, evaluate test results and propose possible explanations for the results if they are unsatisfactory.

Material Resources

The material resources necessary to carry the project are the following:

- *Hardware* needed for the project:
  - *Gigabyte G492-H80 (RTX 3080)*: This will be the HPC equipment necessary for the development and testing of the project. The machine was purchased by the Computer Architecture department in 2022 and is used for courses like TGA integrated in the Boada environment for telematic use through SSH. This will allow to access the hardware from any PC at home. The Computer Architecture department will let the author use the Hardware for the time necessary for the project.
  - *HP OMEN 40L GT21-0022ns AMD Ryzen 5 5600X/16GB/512GB SSD/RTX 3060*: This will be the final PC where the HPC GPU computations will take place because of the risks that turned up to be true and changed the original scope of the project. This PC is perfectly compatible with the same tasks that the Boada environment was going to be used for. The GPU is overall better than the original RTX 3080 for reasons listed in the methodology of the project.
  - *Lenovo IdeaPad 1 15ALC7*: A simple and economic laptop with a *Linux Mint* distro installed will be the main PC to work in the project documentation and programming. Since we only need internet access to get to Boada we can simplify the project material requirements on this end.
- *Software* needed for the project (All free software with no licence needed):
  - *LaTeX(Overleaf)*: For documentation purposes LaTeXoffers a quick and professional documentation environment with lots of libraries that add interesting functionalities.

- *Visual Studio*: A free and powerful code editor that has many extensions that will help with the development of the project.
- *TensorFlow/PyTorch*: The two open source machine learning platforms that will be used to implement the model. The final choice between the two will depend on the initial stages of research that the project will carry out. Both offer options to work with the particular HPC environment that the Thesis will use.
- *GitHub*: For version control and Git usage to keep track of changes and be able to mantain the different versions.
- *Trello*: Online application allows the creation of different objectives in order to manage the different tasks planned on this section.

9.2.2 Tasks Explained

In order to properly explain the project we will separate it in different areas or phases in order to organize the different tasks that need to take place. The division will be centered around the different resources needed for the part.

Project Management (PM) The project management section will mainly involve the *Project Manager* and will revolve around setting up the project and periodically checking and evaluating the evolution through it's life cycle.

- **PM1 - Documentation** will be produced in order to keep track of the work and be able to demonstrate and depict all the progress throughout the project. This task will be a reoccurring one that will be done in parallel through all the project. The task will take an estimate of *80 hours* for all the project. *Overleaf* will be the selected tool for documentation.
- **PM2 - Periodic Meetings** will also be held weekly with the director of the project in order to have validation and to evaluate the evolution of the project. An hour a week will be allocated for this purpose. Along 15 weeks it will cover basically *15 hours*.
- **PM3 - Context and Scope**: Defining the initial context and scope and evaluate the objectives and potential risks. This lays the ground for the whole project to start and defines the different basic aspects of itself. This will take *25 hours* since it needs a very relevant preliminary investigation of the context of the problem and some critical definitions.
- **PM4 - Time Planning**: A specific time planning has to be created in order to ensure success and correct execution of all the different tasks recognised by the manager. This task needs *15 hours* since all tasks have to be defined and the plan has to be feasible.
- **PM5 - Budget**: Precise budget planning will be needed in order to take into account all the costs of the Thesis. Since Deep Learning with HPC can be expensive (both in terms of Hardware and Energy needed) it will be critical to define correctly the Budget needed. For this particular task *10 hours* will be allocated.

- **PM6 - Sustainability Analysis**: In order to keep the Thesis aware of the environment a sustainability analysis will be carried out. Pondering upon the ecological and social implication of the project. This will be important since Deep Learning Training can be costly energetically and AI can have a big impact on society. In this case *5 hours* will be needed for the task.

Research(R) This part of the Thesis will be carried out entirely by the *AI Solutions Architect*. This set of tasks is related to the prior investigation needed for the project to happen.

- **R1 - Deep Learning Convolutional Neural Network Study**: A deeper study of Neural Networks will be carried out in order to gain knowledge on the topic to be able to understand the state of the art to it's full extent and be able to design technologies that are interesting to the academic environment. This process can potentially be cyclical since in latter stages of the project some new aspect of the field can be discovered and investigation on the matter will be needed. Initially *35 hours* of dedication will be surely needed with some variable additional dedication down the line.

- **R2 - State of the art deep evaluation (explore existing implementations)**: With full knowledge of the different theoretical elements that interact in Deep Network models the *AI Solutions Architect* will study the different solutions to gain awareness on the potential design solutions that the project can focus on in order to produce better performances and fully exploit the given Hardware. This task will need *25 hours* of dedication in order to be able to vastly explore the different solutions.

Setup(SE) The setup will mainly cover getting the HPC up and running with everything needed for the code developed to be executed and tested on the hardware.

- **SE1 - Prepare Environment**: Since this thesis will use a HPC that has never been used for AI purposes the environment will need to be set up in order to have all the right dependencies and open-source programs and libraries needed to run. This task can be done by the *Programmer* and will take *20 hours*.

- **SE2 - Select and load dataset**: A certain dataset is already selected but a particular trustworthy source for the dataset will need to be found. This task will take into account *AI Solutions Architect* and can take up to *10 hours* due to the non-trivial nature of uploading such a big dataset to a shared server.

Development (DV)

- **DV1 - Dataset study and preprocessing**: The dataset might have to go through some preprocessing or balancing prior to loading it. This might be related to some natural unbalance between the different classes or some other unforseen problem. This will take *20 hours* because of the complexity and extent of the dataset.

- **DV2 - Design network**: A layer architecture will be defined to create a Convolutional Neural Network that applies all the state of the art known technologies and try to iterate on them. This naturally takes time since a creative process has to take place, taking an estimate of *25 hours*.

- **DV3 - Implement and train network**: Once the layer architecture is designed an implementation can be executed an trained to have a functional version of the desgined network. This task will take an estamate of *30 hours*.
- **DV4 - Parameter tuning**: After testing the initial implementation and evaluating the performance and possible flaws some further parameter tuning takes place in hopes of getting better performances and upgrading the model. This process requires trial and error and educated guesses so it takes some time to get the right setup (*15 hours*).
- **DV5 - Optimize processes**: Further optimizations trying to find low-level optimization through the profiling conducted in the testing. This is a very complex task that will need complex testing results to be compiled and understood to a very close to the hardware approach (taking at least *45 hours* of trying to find improvements).
- **DV6 - Design and implement final network**: Once studied in depth the initial version and iterated a bit through it a final version of the Convolutional Neural Network will be developed taking approximately *30 hours*.

Testing (TS)

All testing tasks will have similar approaches and time estimations and the only thing that will vary between them is the subject of testing so a generic testing procedure will be explained. Firstly, validation of the model will be needed. Testing how good our model performs according to some well known metrics and techniques. After that, a trace analysis will be done in order to see the performance of the model and try to find flaws in the execution (like sequential regions). This will take approximately *10 hours* per test depending on the trace complexity and length of testing reports.

The different tests that need to be carried out are the following:

- **TS1 - Test selected state of the art solution**
- **TS2 - Test and Validate first functional version**
- **TS2 - Test architecture aware version**
- **TS3 - Test final version**

9.2.3 Summary table

Id	Task	Time	Dependencies	Resources	Roles
PM	**Project Management**	**140h**	-	-	-
PM1	Documentation	80h	-	PC	M, P, AI, T
PM2	Periodic Meetings	15h	-	-	M
PM3	Context and Scope	25h	-	PC	M
PM4	Time Planning	15h	PM3	PC	M
PM5	Budget	10h	PM4	PC	M
PM6	Sustainability Analysis	5h	PM5	-	M
R	**Research**	**65h**	-	-	-
R1	Convolutional NN Study	35h	-	PC	AI
R2	State of the art study	25h	R1	-	AI
SE	**Setup**	**30h**	-	-	-
SE1	Prepare Environment	20h	-	PC, HPC	P, AI
SE2	Select and Load dataset	10h	SE1	-	P, AI
DV	**Development**	**165h**	-	-	-
DV1	Dataset Preprocessing	20h	SE2	PC	AI
DV2	Design Network	25h	R2, DV1	PC	AI
DV3	Implement and train CNN	30h	DV2	PC, HPC	P
DV4	Parameter tuning	15h	TS2	PC, HPC	P, AI
DV5	Optimize Processes	45h	DV4	PC, HPC	P, AI
DV6	Implement final CNN	30h	DV5, TS3	PC, HPC	P
TS	**Testing**	**40h**	-	-	-
TS1	Test state of the art	10h	SE2, R2	PC, HPC	T
TS2	Test and Validate 1.0	10h	DV3	PC, HPC	T
TS3	Test and Validate optimal	10h	DV4, DV5	PC, HPC	T
TS3	Test and Validate final	10h	DV6	PC, HPC	T
Total	-	**450h**	-	-	-

Table 11: Table generated in *LaTeX* by the author. There are some abbreviations: M (Project Manager), P (Programmer), AI (AI Solutions Architect), T (Tester/Data Analyst), PC (Lenovo), HPC (Gigabyte G492-H80)

Sequence of Tasks

The sequence of tasks can be seen at table 11. Basically all tasks related to implementation are dependant on both setup and research. Further implementation task depend on the testing of the previous version. Finally, project management tasks are sequential mostly except recurrent tasks like meetings or documentation.

9.3 Estimates and the Gantt

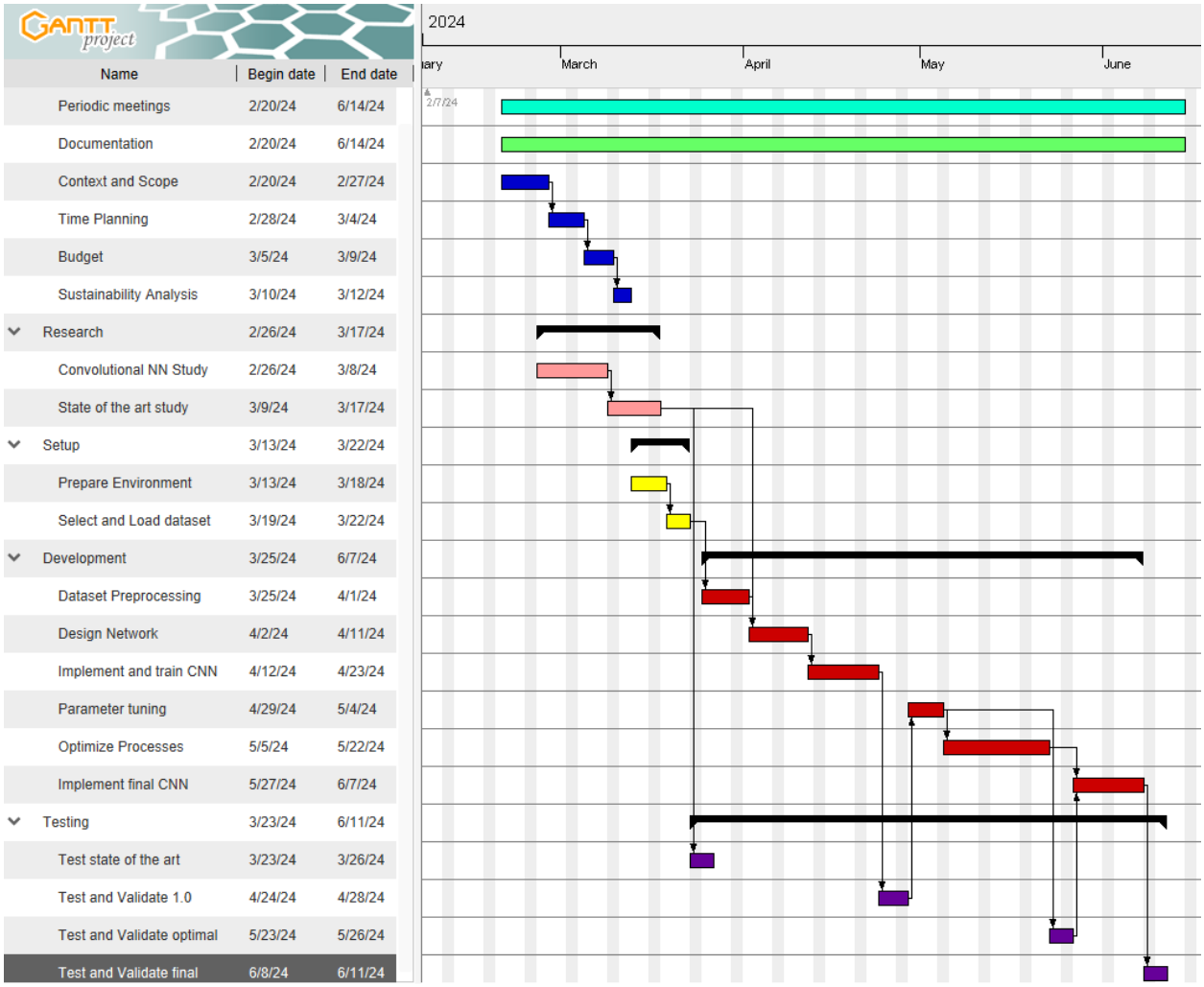


Figure 56: Gantt plot generated with the free software *GanttProject* generated in `https://www.ganttproject.biz/`

The Gantt at Figure 56 the planning of tasks can be seen visually. There's some degree of concurrence achieved in the first parts of the project where Research and Setup can be done simultaneously with Project Management duties. This can't happen later when implementing and experimenting since tasks start to be more sequential. This is due to further implementing being limited by the tests of previous iterations. Some degree of parallelism can be exploited if the designer would design new models as the programmer implements but that is not possible in the context of this thesis.

A daily dedication of four hours is taken into account when creating this Gantt diagram.

The finish-to-start dependencies are portrayed as arrows and tasks are colored according to group of tasks. In the end, the overall structure of the Gantt was followed in a delayed time because of the health issues of the author.

9.4 Risk Management

On this section, mitigation and avoiding strategies will be explored for the obstacles and risks that where introduced in the first delivery.

When talking about obstacles, this thesis implementation being the *first Deep Learning implementaiton* by the author this can be troubling. In order to avoid this obstacle a lot of the wheigh of the project will be put into Research and getting the right Setup for the project so we mitigate any potential notable effects on the duration of the project. That way we give more time on individual tasks in order to accomodate the inexperience on the topic. When researching a special stress will be put into finding implementations and trace studies on similar hardware in order to learn from other's findings. Investigation time will be key to counter that inexperience effect.

One particular obstacle that's more tangible is the memory allocation needed for the project. The Boada computer is shared between plenty of researchers so memory allocation is not easy to get by. In order to avoid facing problems with this particular obstacle swift actions took place and as soon as possible petitions where made to be able to allocate the memory during some short period of time. In case this is not enough to get the proper allocation some undersampling might have to take place. This would minimize undermining the total model performance while saving a lot of space and making the project viable. It would add some *10 hours* of preprocessing but would make possible to carry on with the project with less memory allocation needed. And even for the local PC it's great to be able to run on a smaller dataset to improve efficiency without sacrificing a lot the classification results.

Meanwhile when talking about risks there are plenty of things to take into account.

Firstly, *human error* can have great impacts on the project's feasibility since a big error can result in time loss critical for the pipeline of the project. Furthermore, costly errors can have a significant impact on the project's sustainability profile. In order to mitigate this as much as possible there will be an effort into debugging and good programming practices to minimize the errors. Also testing when getting into grips with the whole environment at first will be carried out with small samples of dataset to not slow down the learning process. The potential impact on project duration can vary from a couple hours of relaunching a program that didn't turn fruitful to whole days (multiple hour processes) if some heavy process like training or validation. For this reason the project is planned to finish 3 weeks ahead of the deadline. Also some code solutions will be implemented in terms of this possible event with checkpoints that can be loaded if power runs out.

To manage tight week schedules with not much free time, time planning has been carried out bearing in mind the existing limitations of the author.

9.5 Budget

In this section a full discussion of the economic cost of the project will be developed in order to quantify costs and gain insight on the expected expenses. The main talking points will be covered are personnel costs, general costs, amortisation, contingencies and incidentals.

9.5.1 Personnel costs

In this section a breakdown of the cost of the work of different people will be found. the tasks described in the Gantt project will serve as a base for the section. Since the different profiles needed for each task have already been assigned and described, this section will define the salary of those professionals. Furthermore, the cost of specific tasks will be defined precisely according to the hours invested by each role on the task.

There are 4 roles involved in the development of the project, some tasks might have more than one person involved in them. Let's get a rough understanding of the general tasks that every role will be related to before the full definition. Firstly, the **Porject Manager** will take all the planning work that needs to be done beforehand (sometimes with the help of more research oriented roles for technical assistance). Secondly, the **AI Solutions Architect** will take part in most of the research and design of the project. This role will also document and draw the main conclusions since this profile carries the most technical knowledge. On other note, the **Junior Programmer** will develop the code necessary as described by the designer. Finally, a **Tester/Data Analyst** will conduct testing and draw insightful conclusions on it.

The hourly rates of each of the roles can be seen at table 12 and how many hours are contributed to each task by all roles is portrayed at table 13. We can see that some broad tasks like Documenting have different roles assigned to it. As well as all development tasks that have both a design and an implementation phase. All tasks have meticulously been broken down into the parts that each particular role will have to work on giving a realistic and precise cost estimate of the human factor of the project. This has also taken into account the critical path in order for the parallel parts to be able to be completed parallel.

All salaries are computed taking into account the work environment in Barcelona. Thus, existing offers have been sorted in order to take out the rough mean annual salaries depicted in the table bellow.

Role	Annual Salary	Pay per hour	Social Security
Project Manager[74]	40.000 €	19.84 €/h	26.78 €/h
AI Solutions Architect[75]	35.000 €	17.36 €/h	23.44 €/h
Junior Programmer[76]	20.000 €	9.92 €/h	13.39 €/h
Junior Data Analyst (tester)[77]	25.000 €	12.40 €/h	16.74 €/h

Table 12: Table generated in *LaTeX* by the author. It portrays the pay per hour for the roles involved in the project development. The annual salary is extracted from the *Glassdoor* website. The pay per hour is calculated dividing the annual salary by all working days in a year (252 days with 8 working hours to be precise)

Id	Task	Total	PM	AI	P	T	cost
PM	**Project Management**	**150h**	80h	55h	-	15h	3,691.7€
PM1	Documentation	80h	10h	55h	-	15h	1808.1€
PM2	Periodic Meetings	15h	15h	-	-	-	401.7€
PM3	Context and Scope	25h	25h	-	-	-	669.5€
PM4	Time Planning	15h	15h	-	-	-	401.7€
PM5	Budget	10h	10h	-	-	-	276.8€
PM6	Sustainability Analysis	5h	5h	-	-	-	133.9€
R	**Research**	**65h**	-	65h	-	-	1523.6€
R1	Convolutional NN Study	35h	-	35h	-	-	820.4€
R2	State of the art study	30h	-	30h	-	-	703.2€
SE	**Setup**	**30h**	-	10h	20h	-	502.2€
SE1	Prepare Environment	20h	-	5h	15h	-	318.05€
SE2	Select and Load dataset	10h	-	5h	5h	-	184.15€
DV	**Development**	**165h**	-	85h	80h	-	3197.5€
DV1	Dataset Preprocessing	20h	-	20h	-	-	468.8€
DV2	Design Network	25h	-	25h	-	-	586€
DV3	Implement and train CNN	30h	-	-	30h	-	401.7€
DV4	Parameter tuning	15h	-	10h	5h	-	301.35€
DV5	Optimize Processes	45h	-	20h	35h	-	937.45€
DV6	Implement final CNN	30h	-	10h	20h	-	502.2€
TS	**Testing**	**40h**	-	-	-	40h	669.6€
TS1	Test state of the art	10h	-	-	-	10h	167.4€
TS2	Test and Validate 1.0	10h	-	-	-	10h	167.4€
TS3	Test and Validate optimal	10h	-	-	-	10h	167.4€
TS3	Test and Validate final	10h	-	-	-	10h	167.4€
Total	-	**450h**	80h	215h	100h	55h	5394.39€

Table 13: Table generated in *LaTeX* by the author. It contains the hour breakdown of the different tasks plus the different costs of every ask.

9.5.2 General costs

Now we will take into account the different generic costs that affect the project on it's entirety.

Amortization

First of all the amortization of the different acquired assets has to be discussed. This project has both hardware and software needs but in this case software is all open source and is not needed to compute the budget.

In order to compute the cost of the Hardware that this project is accountable for, we have to see the amortization. It takes into account the hours that the hardware is used related to the value per hour within the life cycle of the particular hardware. In the particular case of software this observation won't be needed since it's free. The amortization formula[78] is:

$$\text{Cost per hour} = \frac{Cost \times LinearAmortizationRate}{years \times workingDays \times workingHours} \times hoursUsed$$

In table 14 a full breakdown of amortization costs of all the hardware needed is portrayed in order to see how much of the overall cost of the hardware will be amortized on the project. The amortization will be calculated bearing in mind the regulations stated by the BOE-A-2023-25882 [79], which states a maximum of 40% amortization in 5 years maximum for this type of systems.

Hardware	Cost	Life expectancy	Amortization
Gigabyte G492-H80 [18]	8197.75 €	3 years	81.33 €
RTX 3080 (4 units)[80]	859.95(x4) €	3 years	34.125 €
Lenovo IdeaPad 1[81]	628.99 €	3 years	18.72 €
HP OMEN 40L GT21-0022ns[81]	1140.00 €	3 years	33.72 €
Total	13406.54 €	-	167.89 €

Table 14: Table generated in *LaTeX* by the author. It portrays the amortization costs that the hardware used by the project has. The amortization is calculated taking into account (252 days with 8 working hours to be precise). For the Lenovo PC 450 hours will be accounted and for the HPC only the estimated execution time taking into account the tasks that use it (150h). The warranty covers 3 years, therefore that much time will be assumed for the amortization.

Indirect costs

The indirect costs related to the project include those elements that need to be inplace in order for the project to be able to be carried out. In particular, this project is mainly developed remotely from home since the HPC can be controlled through SSH. Because of that the indirect costs are related to the space from which the author decides to work from.

Energy cost

In regards of energy consumption is important to take into account that the project works with a power hungry system such as an HPC.

In order to compute the cost an estimation of the time used will be done. Results can be seen on table 15

Hardware	Power	Time of use	Consumption	Cost
Gigabyte G492-H80 [18]	2200 W	150 h	333 kWh	41,02 €
Lenovo IdeaPad 1[81]	65 W	450 h	29.25 kWh	3,6 €
HP OMEN 40L GT21-0022ns[19]	600 W	150 h	90 kWh	11.09 €
Total	-	-	-	55,71 €

Table 15: Table generated in *LaTeX* by the author. The price of the electrical power is 0.1232 €/kWh at the day of checking[82].

Internet cost

Internet costs at the selected remote work environment are 24.99 €monthly. If we take into account the duration of the project which is 5 months we can conclude the final cost:

$$\text{Cost per hour} = 24.99€/month \times 5months \times \frac{4hoursused}{16hoursadaytouse} = 31.24€$$

Rent cost

The desired workspace is the home of the author. The rent related to the remote workspace is then the rent of the household, in this case a room since the rest of the space can be shared. In the author's area, rent for a room revolves around 300€. Since the project spans over 5 months the total is 1500€.

Summary of General Costs Here we can see the general costs computed in table 16.

Concept	Cost
Amortization	167.89 €
Energy	55,71 €
Internet	31.24 €
Rent	1500 €
Total	1759.84 €

Table 16: Table generated in *LaTeX* by the author. It includes a summary of the general costs of the project.

9.5.3 Possible deviations

Contingencies

Contingencies take into account unforeseen circumstances that can lead to budget adjustments. In order to reduce the potential impact these can have a particular percentage is applied to the final cost in order to cover in case of unfortunate events. In this type of projects 15% of contingency margin is typical. In this case if we take into account both general and personnel costs it amounts to 1066.41 €for contingencies.

Incidental

Now referring to the previously described risks that the project might lead to we have foreseen some potential risks that have to be taken into account. Therefore calculating the potential economic impact and the probability of those particular events taking place. A full explaination of the events won't be necessary since they have already been introduced but in table 17 a full breakdown of the cost that they imply is calculated. For *First AI solution for the architecture*, potentially 15 hours of extra research might be necessary (at a 23.44 €hourly rate). For *Memory allocation for the dataset*, some 10 hours of extra preprocessing

migh be necessary (at 16.74 €hourly rate). Finally for *Human Error*, some 10 hours of bug fixing plus energy from the re-execution are accounted (at 13.39 hourly rate plus 0.12 €per hour of execution).

Risk	Cost	Probability	Final Cost
First AI solution for the architecture	351.6 €	25 %	87.9 €
Memory allocation for the dataset	167.5 €	70 %	117.25 €
Human Error (Faulty Executions on Run)	135.1 €	20%	27.02 €
Total	-	-	232.17 €

Table 17: Table generated in *LaTeX* by the author. It portrays the incidental costs estimated through the potential economic impact and the probability of happening.

9.5.4 Summary Budget

The table 18 contains a brief summary of all budget elements combined generating the final total.

Concept	Cost
Human Resources	5394.39 €
General Costs	1759.84 €
Contingencies	1066.41 €
Incidental	232.17 €
Total	8452.81 €

Table 18: Table generated in *LaTeX* by the author. It includes a summary of all budget elements combined.

9.5.5 Management control

In order to control potential budget deviations, numerical indicators are needed to asses and quantify those differences. Since, despite taking into account contingencies and possible incidents no preparation can assure no failure.

Due to need of knowing by how much the actual process deviates from the planning some models will be discussed and described precisely. In general terms the aim of the model will be to numerically describe the difference between some target observation variable multiplied by modifiers that condition that variable's impact.

The models are the following and focus around the different elements that can vary due to unforeseen planning problems or external forces(economical crashes, electricity shortages etc.):

- Human resources deviation: This deviation model accounts for the potential need of more involvement from the people involved in the project in the form of more hours. Here $estWorkerInTask$ and $realWorkerInTask$ contain the estimated and real hours for a particular worker i in a particular task j.

$$HRD = \sum_{i=1}^{t_{tasks}} \sum_{j=1}^{W_{workers}} ((estWorkerInTask_{i,j} - realWorkerInTask_{i,j}) \times payPerHour_j)$$

- Hardware deviation (Energy + Amortization): This deviation model quantifies the need of more time with the different devices (more critically the HPC). Taking into account in two individual models related to the hours of usage of the hardware. Hadrware Deviation of Energy (HDE) and Hardware Deviation of Amortization (HDA) are defined as it follows:

$$HDE = \sum_{i=1}^{H_{hardware}} ((estHoursUsed_i - realHoursUsed_i) \times (powerConsumption_i \times energyPrice)$$

$$HDA = \sum_{i=1}^{H_{hardware}} ((estHoursUsed_i - realHoursUsed_i) \times \frac{Cost_i \times LinearAmortizationRate}{years \times workingDays \times workingHours})$$

- Incidental cost deviation: This deviation model takes into consideration potential unforeseen incidents. It is modeled in two part (forseen incidents and unforseen incidents) as if follows:

$$ID = \sum_{i=1}^{i_{incidents}} ((estIncidentTime_i - realIncidentTime_i) \times economicImpact_i) +$$

$$UID = - \sum_{j=1}^{u_{unforseenIncidents}} ((realIncidentTime_j) \times economicImpact_j)$$

Thanks to these deviation models, management control can be executed taking close look to the numerical results of them. These results, if negative can lead to management actions to mitigate the potential problems that those indicate and act consequently. The main actions would be related to enlarge the contingency funds to account for the deviations spotted.

9.6 Sustainability report

Self-assesment

I have always been very concerned with sustainability. Given that my generation is going to experiment all the long known and ignored consequences of industrial revolution. Concern and curiosity are natural feelings in those matters. Nonetheless, most of the time, and thanks to an array of political interests and disinformation, it appears a complex matter for one to asses as an individual a clear picture of what sustainability is. More so, to correctly depict the different areas at which it has to be considered and how one can contribute to a greener fair world.

The poll helped in doing a deep self-evaluation on what topics I was more versed and which ones where less present in my mind.

For example, in terms of ecological impact through self exploration it's easy to see how the systems that are around us can impact the ecosystem. In terms of my particular field high performance computing has a big carbon footprint. Luckily, initiatives like the Green500 put a energetic efficiency and ecology on the spotlight. In regards of social impacts I also had a fair amount of knowledge.

On the other hand, social crisis are also closely related to ecological aspects and technological advancements developed in the tech industry. For instance, AI can have a great impact on society taking away jobs of increasing complexity. On a more psychological note, unlimited AI generated content can lead to addiction and abstraction from the real world. Also, the impact on the ecology has a greater impact on the global south manifesting in the form of unbearable climate and ecological disasters.

In terms of economic sustainability, I had basic knowledge on how a poor usage of earths resources and reckless economic activity can lead to big crashes and hard to recover economic situations, but my knowledge on the nuances of economics is more limited.

In terms of general concepts I have a decent coverage, but what surprised me the most was the amount of metrics to quantify and help deal with sustainability at all levels. Social impact metrics where unheard of for me and the poll helped as an introduction to them. Economic viability metrics was another big mystery for me since I have never partaken any economy courses apart from basic ones at college.

Finally, all the topics introduced in the poll are going to lead to a better evaluation of all types of sustainability for my thesis. New concepts will be added to the considerations and already known concepts will be reinforced through a new more complete lens.

Economic Dimension

Regarding PPP: Reflection on the cost you have estimated for the completion of the project

A full cost estimation has been done for the thesis at section 9.5 that takes into account

a realistic depiction of a full team of 4 members. Moreover, it contains estimations for both contingencies and incidental costs for the unforeseen problems. Finally, it contains management control metrics to evaluate the budget.

The resulting budget has a very reasonable cost of 8408 €. This cost with all the considerations done seems realistic and feasible. If the right funds are gathered through investors the project would be able to carry on with the stated resources.

Regarding Useful Life: How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)?

Mainly, the economic issue related to the problem currently is efficiency. Bearing in mind that Deep Learning is very costly in resources to train and deploy.

In order to minimize the impact of the project, starting from existing models potentially pre-trained might be interesting in order to conduct research on a low impact profile.

How will your solution improve economic issues (costs ...) with respect other existing solutions?

This project's main point is to create a model that's efficient for the given machine. This will impact on the economic issues related to the implementation on similar machines. Since the project can be selected as a base for future models without the economic resources needed to get to that point of efficiency and accuracy.

Environmental Dimension

Regarding PPP: Have you estimated the environmental impact of the project?

A particular estimation has not been carried out but the basic principles of environmental impact have been considered. On the one hand, the energy consumption for the whole execution of the project is taken into account both to control the budget allocated to it but also to get a grasp on how much energy are we consuming resulting in enlarging the carbon footprint of the project. On the other hand, hardware will be used but is also reusable by many students after the thesis is done. In fact currently is being used for a course at FIB called TGA (Graphics Cards and Accelerators).

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

As introduced earlier, the hardware resources used are shared and plan to be used by the educational organization for a long time past their official warranty. In that sense, the carbon footprint produced by the creation of the hardware will be amortized for the longest time possible letting students interact with powerful hardware with little impact on the planet.

Moreover, software resources will be reused too taking into account previously trained models as a starting point for the model developed.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?

In terms of ecological aspects of the useful life the principles mentioned in the economic part can be applied. Since larger power consumption will result in a larger carbon footprint. In this particular case for such a big dataset it is critical to not waste a lot of energy training the same models again and again so contemplating existing solutions is vital.

How will your solution improve the environment with respect other existing solutions?

With greater efficiencies lower power consumption would be needed both in training and deployment. Thus creating a solution that will improve the environmental aspect of the general solutions found that focus more on the accuracy.

Social Dimension

Regarding PPP: What do you think you will achieve -in terms of personal growth- from doing this project?

Personally, I think this project will allow me to do a deep dive on a topic that I am very interested in and haven't seen trough the course of my degree. Having designated time to do full research and form knowledge autonomously will be a great opportunity to keep an active learning process past graduation. Finally, having a large project in hands by myself will upgrade my autonomous skills and force me to be more proactive with others.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?

Currently the problem that I want to address is solved with human intervention. Since the model doesn't yet have an accuracy that can ensure successful autonomous diagnostic.

How will your solution improve the quality of life (social dimension) with respect other existing solutions?

If the project is successful and a significant efficiency improvement can be found this could help with large scale diagnostics. This would help in preventive cases where mass diagnostics are done to catch potential illnesses soon. This would have a great impact on society providing a good service without taking jobs from anyone since it would only act as a precautionary layer that would in the end lead to human professionals.

Regarding Useful Life: Is there a real need for the project?

There are a lot of existing solutions that are competent and accurate but implementations on small hardware that can be acquired by health organizations in order to do their own research and diagnostics locally is interesting. Also if enough efficiency is found so that a new frontier on the state of the art is discovered it would be a great contribution.

10 Conclusion

10.1 Objectives and Findings

To conclude the project, we can state that the general goal of the project of investigating and producing a software solution for the proposed problem of multi-label classification for the NIH X-Ray dataset. The resulting solution is the product of the iterative process carried out during the project. The different objectives of the project where met incrementally producing checkpoint models architectures with increasing complexity, efficiency and classification capability.

In terms of investigation, the resulting documentation contains all the necessary knowledge related to the project in terms of background from a initiate perspective so the different conclusions and investigations undergone are accessible and properly contextualized. This contextualization initially contains a simple introduction to Deep Learning through the lens of a Computer Scientist. In addition, specific background references to technologies related to more specific problem needs such as imbalance management and advanced learning rate schedulers where explored. As well as an explanation of the different labels of the classification task with a medical background in order to better inform the researcher on the particularities of the dataset. Exploration of the dataset with different techniques was done in order to understand the needs of the dataset to adapt our model to it.

As a result of the exhaustive background investigation and documentation, a varied and complete experimentation was undergone with different successful implementations. These made the final architecture significantly better than the first iteration both in performance and classification terms. In terms of experiments we selected the different technologies that Deep Learning has to offer in order to select the best fitting. Expanding on previous state of the art implementations with new papers that could introduce helpful functions.

In terms of optimizers we tested all alternatives finding that Adam is the best but in relative terms since SGD can have it's advantages when paired with adequate learning rate scheduling. Since when parining SGD with a fluctuating learning rate it was found that Cyclic learning with proper hyperparameters ensures a more steady convergence with better overall results. The resulting solution also takes into account the inherent imbalance of medical data with a the Asymmetric Loss Function inherited from other studies. This function wasn't tested with this particular problem by other studies and the results where very positive. In terms of model selection we tested the EfficientNet achieving better results in a fraction of the time that models like DenseNet121 needed. In terms of computational performance, some hyperparameter study of the data loading functions was made in order to find a better parallelisation for the pipeline. Our final performance has exceeded the original 2017 implementation of the ChestXRay dataset provided but doesn't achieve the absolute state of the art. Since our best AUCs of ROC curves are 0.822 and state of the art can achieve 0.847 at the moment with some technologies that fall out of scope out of complexity [14].

Finally, the source code for the software solution is offered as *open-source MIT licensed* code in GitHub in order to have the findings be of use to other researchers so that the state of the art can advance further. All functional and non-functional requirements where met in order to produce a solution that is easy to use and sufficiently powerful to investigate with. Code is well organized in order to be *maintainable* and through profiling we proved our code is *analyzable*, producing an *efficient* solution with its best configuration. Plus, the GradCAM integrated with the model makes the architecture more *transparent*.

In terms of academic and personal development for the author this project has been very fruitful in the acquired knowledge and significant experience in technologies of particular interest. Since deep learning's spotlight is relatively recent most of the academic options for the author at the time didn't include proper introduction to such themes. Learning through historic papers and understanding each breakthrough to produce a code solution has been an enriching experience that will allow the author to place the first stepping stone on a research career related to AI.

In terms of challenges, a part from the time schedule problems related to personal events, the investigation ended being a particular hurdle. Since Deep Learning is such a broad topic it is easy to lose one self into the rabbit hole of research. Such elongated research produced a delayed start of the implementation efforts which eventually produced a tight deadline. Code solutions of other project where abundant but complex to pick up and iterate from so the implementation was made from zero. Most of the core elements where personally coded except inherited functions from cited open-source projects which ended up taking more time than expected.

10.2 Future Work

Our final model is fairly competent but it could benefit from some more investigation in the future.

- *Early stopping problems*: The model could benefit on some deeper investigation in terms of early stopping. Since Learning Rate Schedulers like Cyclic have allowed for better performance but the model still stops in a relatively early 20 epochs when it should be able to improve for a longer period of time.
- *Cluster HPC architecture*: Our implementation with conda should be easy to use in higher grade clusters with multi-GPU like boada and as a future project it would be interesting to run more demanding alternatives of EfficientNet with higher image resolutions to see if it could shave a bit more Validation Loss from the training process.
- *Potential Overfitting Treatment*: Some models have been proven to produce some degree of overfitting. It would be interesting to find a way of minimizing it.
- *Better data balancing methods:* Data level imbalance management is trivial in this project and the training process could benefit of a balanced validation set that prioritizes all classes equally. This should be done with some smart solution like SMOTE or some type of oversampling of under represented variables.

References

[1] M.A. Boden. *Mind as Machine: A History of Cognitive Science*. Clarendon Press, 2008.

[2] Albert Njoroge Kahira. *Convergence of Deep Learning and High Performance Computing: Challenges and Solutions*. PhD thesis, Universitat Politecnica de Catalunya, Barcelona, Spain, July 2021. Dissertation submitted to the Department of Computer Architectures (DAC) in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Architectures.

[3] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S. Vetter. Nvidia tensor core programmability, performance i& precision. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 522–531, 2018.

[4] S. Kevin (editor) Zhou, Hayit (editor) Greenspan, and Dinggang (editor) Shen. *Deep Learning for Medical Image Analysis*. Academic Press, 2023. The MICCAI Society book Series.

[5] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3462–3471, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[6] Facultat d'Informàtica de Barcelona. Competences, Accessed 2024. Content retrieved from the website of the Facultat d'Informàtica de Barcelona, accessed in 2024.

[7] Joseph Yacim and Douw Boshoff. Impact of artificial neural networks training algorithms on accurate prediction of property values. *Journal of Real Estate Research*, 40:375–418, 11 2018.

[8] Andrew Glassner. *Deep Learning: A Visual Approach*. No Starch Press, illustrated edition, 2021.

[9] Computer Vision Foundation. `https://www.thecvf.com`. Accessed: February 25, 2024.

[10] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning, 2017.

[11] Laleh Seyyed-Kalantari, Guanxiong Liu, Matthew McDermott, Irene Y. Chen, and Marzyeh Ghassemi. Chexclusion: Fairness gaps in deep chest x-ray classifiers, 2020.

[12] Constantin Seibold, Simon Reiß, M. Saquib Sarfraz, Rainer Stiefelhagen, and Jens Kleesiek. *Breaking with Fixed Set Pathology Recognition Through Report-Guided Contrastive Training*, page 690–700. Springer Nature Switzerland, 2022.

[13] ImageNet Research Team. ImageNet. https://www.image-net.org, 2024. Accessed: February 25, 2024.

[14] Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Multi-objective evolutionary design of deep convolutional neural networks for image classification, 2020.

[15] Gregory Holste, Song Wang, Ziyu Jiang, Thomas C. Shen, George Shih, Ronald M. Summers, Yifan Peng, and Zhangyang Wang. *Long-Tailed Classification of Thorax Diseases on Chest X-Ray: A New Benchmark Study*, page 22–32. Springer Nature Switzerland, 2022.

[16] Paraver. https://tools.bsc.es/paraver, 2024. Copyright © 2024, tools [at] bsc.es, Theme by BSC Tools.

[17] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.

[18] Broadberry Data Systems. Gigabyte g492-h80 server specifications. https://www.broadberry.eu/intel-xeon-sp-g3-gigabyte-servers/g492-h80-gigabyte-servers, Accessed 2024.

[19] Hp omen 40l gt21-0022ns amd ryzen 5 5600x/rtx 3060. https://www.pccomponentes.com/hp-omen-40l-gt21-0022ns-amd-ryzen-5-5600x-16gb-512gb-ssd-rtx-3060, 2024. Accessed: 2024-10-07.

[20] Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification, 2021.

[21] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[22] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969. Discusses limitations and flaws of perceptrons, especially in handling non-linearly separable data like XOR.

[23] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[24] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. pages 75–78, 09 2019.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[26] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation

network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[31] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

[33] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[34] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[35] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024.

[36] Ronald (NIH/CC/DRD) Summers. ChestXray-NIHCC Dataset. https://nihcc.app.box.com/v/ChestXray-NIHCC, 2017. Creado el 1 sept 2017, 20:33. Última modificación el 11 sept 2023, 17:31.

[37] Ronald (NIH/CC/DRD) Summers. Readme_chestxray. https://nihcc.app.box.com/v/ChestXray-NIHCC/file/220660789610, 2018. nov 5th 2018.

[38] National Center for Biotechnology Information. Ncbi homepage. https://www.ncbi.nlm.nih.gov/, 2024. Accessed: 2024-09-06.

[39] K. Grott, S. Chauhan, D. K. Sanghavi, and et al. *Atelectasis*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2024 Feb 26].

[40] JH Woodring and JC Reed. Types and mechanisms of pulmonary atelectasis. *Journal of Thoracic Imaging*, 11(2):92–108, 1996.

[41] A. Cortés Campos and M. Martínez Rodríguez. Manifestaciones radiográficas de las atelectasias pulmonares lobares en la radiografía de tórax y su correlación con la tomografía computarizada. *Radiología*, 56(3):257–267, 2014.

[42] Graham Lloyd-Jones. Chest x-ray - lung cancer: Mass vs consolidation. https://www.radiologymasterclass.co.uk/gallery/chest/lung_cancer/mass_consolidation, 2019. Last reviewed: October 2019.

[43] Sherif Assaad, Wolf Kratzert, Benjamin Shelley, Malcolm Friedman, and Albert Perrino. Assessment of pulmonary edema: Principles and practice. *Journal of Cardiothoracic and Vascular Anesthesia*, 32, 08 2017.

[44] SherifA Nasef, AliM AlMuslim, AmerS Zahralliyali, Hisham Almomen, and AlsirG Ali. A case of rhuematoid arthritis associated with constrictive pericarditis. *International Journal of Advanced Research*, 5:1106–1109, 12 2017.

[45] H. Amin and W. J. Siddiqui. *Cardiomegaly*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2022 Nov 20].

[46] R. Krishna, M. H. Antoine, M. H. Alahmadi, and et al. *Pleural Effusion*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2024 Aug 31].

[47] P. Pahal, A. Avula, and S. Sharma. *Emphysema*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2023 Jan 26].

[48] E. Yu and S. Sharma. *Cystic Fibrosis*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2022 Aug 8].

[49] K. Spellar, S. Lotfollahzadeh, and N. Gupta. *Diaphragmatic Hernia*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2024 Jul 23].

[50] R. Garcia-Carretero, O. Vazquez-Gomez, B. Rodriguez-Maya, and et al. Delayed diagnosis of an atypical pneumonia resembling a solitary pulmonary nodule. *Cureus*,

13(11):e19456, November 10 2021.

[51] A. Wyker, S. Sharma, and W. W. Henderson. *Solitary Pulmonary Nodule*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2024 Aug 12].

[52] A. Saito, Y. Hakamata, Y. Yamada, M. Sunohara, M. Tarui, Y. Murano, A. Mitani, K. Tanaka, T. Nagase, and S. Yanagimoto. Pleural thickening on screening chest x-rays: a single institutional study. *Respiratory Research*, 20(1):138, July 5 2019.

[53] V. Jain, R. Vashisht, G. Yilmaz, and et al. *Pneumonia Pathology*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2023 Jul 31].

[54] C. L. McKnight and B. Burns. *Pneumothorax*. StatPearls Publishing, Treasure Island (FL), 2024. [Updated 2023 Feb 15].

[55] Athanasia Pataka, Cristine Radojicic, Mathina Darmalingam, and Ioannis P. Kioumis. Assessment of persistent pulmonary infiltrate images. `https://bestpractice.bmj.com/topics/en-gb/1094`, 2024. Last reviewed: 9 Aug 2024; Last updated: 25 Jun 2024.

[56] Jan de Leeuw. Principal component analysis of binary data by iterated singular value decomposition. *Computational Statistics Data Analysis*, 50:21–39, 01 2006.

[57] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[59] Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th International Conference on Machine Learning*, pages 935–942, Boca Raton, FL, USA, 2007. Florida Atlantic University.

[60] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[61] Inc. Anaconda. Anaconda: The operating system for ai. `https://www.anaconda.com/`, 2024. Accessed: 2024-10-07.

[62] National Institutes of Health (NIH). Nih clinical center provides one of the largest publicly available chest x-ray datasets to scientific community, 2017. Accessed: 2024-09-16.

[63] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[64] S.M. Nabil Ashraf, Md. Adyelullahil Mamun, Hasnat Md. Abdullah, and Md. Golam Rabiul Alam. Synthensemble: A fusion of cnn, vision transformer, and hybrid models for multi-label chest x-ray classification. In *2023 26th International Conference on Computer and Information Technology (ICCIT)*, page 1–6. IEEE, December 2023.

[65] Alin Stein. X_ray: A tool for x-ray image classification. `https://github.com/alinstein/X_RAY`, 2024. Last Accessed: 2024-10-07.

[66] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[67] Geoffrey Hinton. Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude, 2012. Coursera: Neural Networks for Machine Learning.

[68] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[69] Leslie N. Smith. Cyclical learning rates for training neural networks, 2017.

[70] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[71] Katsura. Pytorch cosine annealing with warmup. `https://github.com/katsura-jp/pytorch-cosine-annealing-with-warmup`, 2023. Last Accessed: 2024-10-07.

[72] . Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44, 09 2020.

[73] Mengdi Huang, Chetan Tekur, and Michael Carilli. Introducing native pytorch automatic mixed precision for faster training on nvidia gpus. *PyTorch Blog*, 2020. Accessed: 2024-10-11.

[74] Glassdoor. Project manager salaries, 2024. Accessed on March 10, 2024.

[75] Glassdoor. Artificial intelligence engineer salaries, 2024. Accessed on March 10, 2024.

[76] Glassdoor. Junior programmer salaries, 2024. Accessed on March 10, 2024.

[77] Glassdoor. Data analyst salaries, 2024. Accessed on March 10, 2024.

[78] Departament d'Organització d'Empreses, Facultat d'Informàtica de Barcelona (FIB). GestiÓn de proyectos: Módulo 2: Aspectos básicos de la gestión de proyectos - 2.4. gestión económica. PowerPoint presentation, Year Presented. Presented at Universitat Politècnica de Catalunya, Facultat d'Informàtica de Barcelona.

[79] Ministerio de Hacienda y Función Pública. BoletÍn oficial del estado, 2023. Official Gazette of the Spanish Government, Núm. 304, Thursday, December 21, 2023, Sec. I. Pág. 168621.

[80] Coolmod. Gigabyte geforce rtx 3080 gaming oc lhr 10gb gddr6x - tarjeta gráfica, 2024. Accessed on March 10, 2024.

[81] PC Componentes. Lenovo ideapad 1 15alc7 amd ryzen 5 5500u/16 gb/512gb ssd/15.6", 2024. Accessed on 10th March 2024.

[82] Jaime. ¿cuál es el precio de la luz hoy y las horas más baratas?, 2024. Updated on March 8, 2024.