RadIA-Logique

A weakly supervised learning strategy
for the detection of pathologies in CXR images

Jonathan Beaulieu-Emond,20101842

CIUSSS Centre Sud de l'île de Montréal,
400 Boul. Maisonneuve Ouest
Superviseur : Jean-Francois Samson,
jean-francois.samson.ccsmtl@ssss.gouv.qc.ca

Département d'informatique et de recherche opérationnelle
Université de Montréal
Canada
February 23, 2023

Contents

RadIA-Logique : A weakly supervised learning strategy for the detection of pathologies in CXR images

February 23, 2023

Abstract

Chest X-ray images (CXR images) require the evaluation by a radiologist before the main attendant for the patient can move forward with a diagnosis and treatment. However, there is currently a shortage of personnel, which leaves the radiologists left to be quickly overworked. This lead to a deterioration in their performance , and longer waiting times.

Currently, radiologists will give about 15 seconds of their time per images in order to review the presence or absence of a multitude of pathologies. This, combine with the limited information they have to work with, leads to a rather high error rate [1] that could be mitigated by the use of deep learning tools.

We built upon the work of this team at Stanford, following their publication [2], [3] to develop our own deep learning model to identify a specific selection of pathologies. I managed to obtain similar results of the CheXpert competition, achieving an average AUC of 0.82 with a single model , compare to the 0.896 achieved by the JFhealthcare team [4]. While it might seem far off, our choice of model was driven by the F1-score, and therefore does not select the optimal AUC score. This combine with the fact we were focusing on more classes (12 instead of only 5 in the competition) might eplains some of the difference. Also, we did not applied a thorough hyperparameter tuning, which could have lead to better results. Using the f1-score as a comparison, we achieved an average of 0.59, compared to the human average of of 0.615 [1]. This means we are within 4% of a radiologist performance

On the demand of our radiologist, we identified 14 classes to identify the presence of in CXR images. I then trained our model on both private data gathered from the patients of the CCSMTL and public data made available by different research group across the planet. Results on our own dataset were inconclusive and will most likely require the help of a radiologist to be properly evaluated, and to clean the data. When validated on our private dataset, we obtained an AUC of 0.77 on average, and a F1-score of 0.43

1 Introduction

Hospitals are now facing a lack of resources and personnel. While sometimes, doubling down on recruitment effort seems like the only way to attack this problem, automating some of their task , or having automated system help the staff might present itself as an alternative solution. More specific to our case, radiologists are well sought after by hospitals. There is currently a national shortage of radiologist in Canada [5], and the situation is not expected to improve in the near future.

This results in overworked radiologists who , in turn, perform worst and have little to no time to review a patient file before giving their opinion. After talking with the radiologists from the Notre Dame hospital, they mentioned reviewing the images of a patient for an average of 15 seconds only, while trying to identify lesions, fractures, infections, masses, etc.

It is in this context of radiologist shortage that the CIUSSS Centre Sud de l'île de Montréal (CCSMTL) has decided to launch a pilot project in artificial intelligence. It has been demonstrated by a team of researcher at Stanford that a deep learning model could be used to classify Chest X-ray images of the lungs by the presence or absence of a multitude of pathologies [2], [3]. For their CheXpert paper, the Stanford team released a dataset that contains 224,316 chest radiographs of 65,240 patients as part of an online competition. Multiple teams participated in this online challenge , with three deserving special interest for their results as seen in 1. The team that won the competition, CheXpert, was able to achieve an AUC of 0.93

After reviewing these, and having a quick meeting with two of our radiologists, we identified a list of 14 pathologies they would like us to identify. Based on the Hierarchical learning paper from the Vingroup Big Data Institute [6] , we have decided to organize our pathologies in a similar hierarchical order, in case we might be able to use it for either the training or inference of our model just like they did. (see figure 21 in the appendix for the details). They have also asked us to try to identify if catheter and tubes inserted into a patient were placed correctly, but this objective has been moved to another model/project .

On the same meeting, our radiologist revealed they wanted to see different goals to this project. They mentionned they weren't interested in the exact details of the pathologies, meaning we would be able to give them only the main parent class, potentially improving our results. While we still believe that the identification of the specific pathology could improve the quality of care provided to the patients of the hospitals, the radiologist we talked to was, for now, reluctant to the idea of a machine being able to performed at an equal level . They therefore have asked us to highlight the problematic region of the image and to let them decide the final diagnosis. Our problem has therefore changed, from a supervised classification one, to a weakly supervised object detection problem.

Rank	Date	Num of Rads below curve	Avg. AUC	Paper's name
1	Aug 31,2020	2.8	0.930	Large-scale Robust Deep AUC Maximization: A New Surrogate Loss and Empirical Studies on Medical Image Classification
2	Sep 01,2019	2.6	0.930	Hierarchical-Learning-V1 (ensemble) Vingroup Big Data Institute
5	Oct 10,2019	2.8	0.929	YWW(ensemble) JF&NNU

Table 1: Extract of the CheXpert classification table [3]. "Num of Rads below curve" refer to the average number of radiologist out of the three that the model beats when it comes to classification

1.1 CheXnet

Our project is based on the recent development by a Stanford team on the CheXnet [2] project and its follow-up,CheXpert [3]. These two projects had one common goal : to develop an AI able to help, or in time maybe even replace radiologists for the analysis of Chest X-Ray (CXR) images. Their first project , CheXnet, was based on a deep learning model that was able to predict 14 different diseases from a CXR image. While they did manage some interesting results, many criticism arose from the medical community. One such critic, the Dr. Lauren Oakden-Rayner, a radiologist and a Senior Research Fellow at the Australian Institute for Machine Learning, did a deep dive into the project, going as far as having a back and forth with the author to help them publish edited version of their paper. As these criticism were quite interesting and led to the follow-up project, we will go over them in the next section.

1.2 Criticism

Dr. Oakden-Rayner did a deep dive into the project, and before going into the CheXpert project, did a good recapitulative of the issues with the CheXnet project [7]. In her own word :

1. Variability: Lots of very similar cases, because there were many patients who had numerous studies (i.e., repeat ICU films). For example, while only 7% of patients had more than 10 films, this still made up 45% of the total dataset. This means that while the overall number of films in the dataset is impressive, the variability is equivalent to a much smaller dataset (we might call this the effective size of the dataset).
2. Labelling method: Labelled via natural language processing, which both has an error rate as a method, and an irreducible error due to the fact that reports don't actually describe images very thoroughly.
3. Labelling quality: Labels didn't seem to match images very well, on the order of 30-90% error rates for the various classes.
4. Label structure: Some of the labels were very difficult to interpret, with a range of labels that describe pretty much the same visual appearances (like "consolidation", "pneumonia", "infiltration").
5. Hidden stratification: Some of the labels contained clinically important subgroups (strata) that were not labelled, for example the pneumothorax (collapsed lung) class didn't distinguish between deadly untreated pneumothoraces and completely safe, well-treated pneumothoraces. These subsets seemed to lead to models that learned useless and potentially dangerous things, like only identifying treated pneumothoraces and missing untreated ones.
6. Documentation: The CXR14 paper, and the additional documentation (here), do not adequately describe the data. It is unclear how the labels were defined, what the cohort characteristics are, or how the dataset could reliably be used.
7. Image quality: The images in the dataset have been downsampled (from 3000 x 2000 pixels to 1024 x 1024) as well as having heavily reduced the number of grey-levels (normally between 3 and 4 thousand, now only 255). This severely harms interpretation of many conditions. Of note, subtle pneumothoraces, small nodules, and retrocardiac opacities become nearly impossible to diagnose for a human expert.

The full detail of the criticism can be found in the blog post [8].

During the project, I've tried keeping in mind all these mentions and I will cover most of them in the future sections. But already, we can realize the limitations that arise from the CheXnet project. Their automated labeler, combined with the usual error rate of a radiologist, meant that by simply splitting their dataset into the training, validatation , they necessarily included a lot of noisy label into each categories. This means that, not only the model might have a harder time training on those noisy labels, but also that we will have difficulty in evaluating the model on the validation set. This resulted in a model that underperformed, obtaining an average F1-score of 0.435, and an average AUC of 0.84. They did however, manage to build a test set , with the help of four radiologists to review the images and labels.

While there exists techniques to train a model adequately with noisy labels, it is crucial to have a validation set that is as close as possible to the test set, as to allow for the selection of the best model possible. This is why the CheXpert project was born.

1.3 CheXpert

The second project, CheXpert, was based on the same model but with a larger dataset (that the Stanford team curated themselves) and a more precise analysis of the diseases. While they kept the 14 original diseases, they mainly focus on 5 different diseases from a CXR image (Atelectasis,Cardiomegaly, Consolidation,Edema and Pleural Effusion). It corrected a few of the criticsm from the last project, and had a different goal. While the Stanford team focused on building the deep learning model for the CheXnet project, which used publicly available data from the National Institute of Health (NIH), which was criticised for being often mislabeled, this project focused on building a new dataset. They gathered over 200 000 images from the Stanford hospital for the project, and while most label were derived from an automated labeler (to convert the text annotation of the radiologists in the patient's file), which assigned a positive, negative , or uncertain flag to each of the label for a given report. The validation and test dataset reviewed by a panel of three radiologists to ensure the quality of the dataset. This is a necessary steps as the evaluation of CXR images is a very complex task, with not always clear answers. In their CheXnet project, they evaluated the performance of their radiologists to be at around 0.4 with the F1-score [1] . This is a rather low score, and therefore explains the need to have a validation/test set reviewed by a panel of radiologists.

While most of the dataset was released, they kept the test set private to evaluate the performance of models submitted to them for the CheXpert competition they created. This competition was launched in 2019 and is still ongoing. At the moment of writing this report, the best model has an AUC of 0.93, a score achieved by both the Big Data Institute[9] and the Deep AUC maximization model [10], which is a very impressive score.

1.4 Our project

For our project, while we mostly worked with the CheXpert dataset, we also started building our own, using data from the different hospitals covered by the CIUSSS Centre-Sud-de-l'Île-de-Montréal (CCSMTL) . Sadly delays in the acquisition of the data and the lack of a clear protocol for the processing of it meant that we were only able to start working on it fully by September.

We also had to deal with the fact that this data was not labelised yet. We received the image in the DICOM format and had to convert them to a more usable format (jpeg). The images were

[1] As mentionned in Dr. Oakden-Rayner blog [8], this was most likely underestimated. ChexZero [1] provides a possibly more accurate estimate

linked with a text report from the radiologist, which we had to convert to a label [2]. Since our text were in french, and that most ressources available were in english, we had to develop our own rule-based system to convert the text to a label. This part of the project was done by another student, Maxime Fournier, and will therefore not be discussed in details here.

Once the data was ready, we separated the dataset into a training, validation and test set, with 300 patients per classes in the validation set (for a total of 9500 images), and 300 (20 images per classes) in the test set. We ensured there were no patient leakage between the different sets(a patient seen during training should not have images present within the test or validations set) [3].

Sadly, the CCSMTL was not able to provide us with expertise in the form of radiologists to review the dataset, or our conversion from text to label. We therefore had to rely solely on our interpretation of the text and the opinion given by the one radiologist who initially reviewed the images. This is a major limitation of our dataset, and might explain the poor performance of our model later on. At the moment of writting[4], we were still waiting for the CCSMTL to provide us with a second meeting with the radiologists, in order to hopefully secure time to have them review and validate our test dataset [5] . Please also note that most of our analysis will be conducted on the CheXpert dataset, both as it has been validated by radiologists, and since the data from the CCSMTL was not ready yet [6].

As mentionned before, we had to take care of the conversion from text to label. While it's another intern who mostly worked on that part, I will briefly described a few of the issues encountered with the help of anonymized examples (they are in french and I will not translate them as to avoid errors that this translation could introduce)

1.4.1 Uncertainties

> Possible petit épanchement à la base droite. Pas d'oedème pulmonaire définitif. Pas de pneumonie définitive. Légère rotation vers la droite ce qui pourrait possiblement expliquer un aspect plus dense du hile droit. Un contrôle est suggéré pour exclure une pathologie hilaire droite.

As you can see, this text is quite ambiguous. It could be a pleural effusion, and the mention "Pas d'oedème pulmonaire définitif" could either mean there is definitely no pulmonary edema, or that there is no "Oedème pulmonaire définitif". The same goes for the mention of a possible pleural effusion, which could either mean that there is a pleural effusion, or that there is a possible pleural effusion. This ambiguity is a major issue, and we had to decide how to handle it. We identified the mention of a disease, and thanks to a rule base classifier determined whether the mention was positive, negative, or uncertain. This however just pushed the uncertainty issue further down the line, and comes as an added error source.

1.4.2 Specific exam

> Renseignements clinique: Fièvre. Éliminer pneumonie. POUMONS: Il n'y a pas d'épanchement. Le médiastin est sans particularité. Il y a une opacité mal définie

[2] The details of this will be covered later.

[3] As per the criticsm of Dr. Oakden-Rayner

[4] 2023-01-03

[5] While it would be very important to also have a validation dataset validated by radiologist, the CCSMTL was reluctant to try and provide us with the ressources to do so. We are currently in saynegotatiation as to obtain time with the radiologist to review data for a test set only.

[6] The data was only fully delivered to us towards the end of the internship

en péri-hilaire droit qui n'était pas présente à l'examen antérieur et qui pourrait donc être une pneumonie. CONCORDANCE IMPOSSIBLE

Again, we see some ambiguity in the text, with the mention of the opacity. However, the issue I want to show here is with the clinical mentions. The radiologist was asked to looked for a pneumonia, and the patient had fever. This is a specific exam, and the radiologist was therefore not looking throughly for other signs of diseases, as they on average only spends about 15 seconds on each image [7].

1.4.3 Reference to prior exams

Image pleuro-parenchymateuse et silhouette cardio-médiastinale sans particularité, notamment sans cardiomégalie identifiée. La silhouette de l'aorte est relativement inchangée. À noter qu'au moment de l'i nterprétation, la patiente avait déjà bénéficié d'une tomodensitométrie. S'il vous plaît vous référer au rapport de cette étude.

As can be seen, when radiologists review a patient's file, they often open previous exam to compare their most recent X-rays to the previous ones. They will sometimes mention this, saying there is no change , but this comment alone does not inform us about the presence or absence of the pathology, and we do not have access to the rest of the patient file to give context to this comment. Also, radiologist usually spell out their report, recording it vocally such that a secretary will later type it, without the radiologist always reading it back. This leads to two other sources of error in our data. Please also note the mispelling of the word "i nterprétation" , with such misspelling possibly misleading our rule-based classifier

1.4.4 Poor image quality and no relecture

Les coins inférieurs des deux poumons ne sont pas entièrement couverts dans la présente radiographie. Il y a progression d'opacités parenchymateuses mal définies au tiers moyen/inférieur du poumon gauche, très discrètement également au tiers moyen/inférieur du poumon droit. Ceci est compatible avec des infiltrats infectieux, possiblement en lien avec COVID. Possible légère surcharge. Pas d'épanchements pleuraux significatifs. Légère cardiomégalie connue. Signé sans relecture du radiologiste.

As shown here, even the radiologist will sometimes comment on the quality of the image, such as partial images.

All these issues are major sources of error in our dataset. While we did try our best to mitigate them, we are aware that they are still present, and that they might have a negative impact on our results. We already mentionned multiple times the need for a second meeting with the radiologists, and we hope that this will allow us to mitigate some of these issues.

Please note our data was anonymized and as such we cannot provide the name of the radiologist, whose report we quoted here. We are also forbidden from providing the images linked with those reports.

These problem, while not completely unique to us , do differ from the ones encountered by other teams building similar dataset. such an example (taken from MimicCXR [11]) is shown here 1. Their radiologist's report seem to be more precise than ours, and more detailed which might lead to issues training our model later on. Sadly, without the help of a trained radiologist,

[7] This specific timing was described to us by the radiology team of the Hôpital Notre Dame

Section	Report	Label
Impression	No evidence of acute cardiopulmonary process.	No Finding
Findings	The left lung is relatively well aerated and clear. The right hemithorax is markedly opacified with volume loss, circumferential pleural thickening and pleural fluid with near complete opacification of the right lung with right basal pleural catheter noted. Hydropneumothorax previously seen is not as well evaluated on this not fully upright film. Cardiac contours are somewhat obscured but unremarkable.	*No Cardiomegaly* *No Enlarged Cardiomediastinum* Pneumothorax[u] Airspace Opacity Pleural Effusion Pleural Other Support Devices
Other	Cardiac size is top normal. Bibasilar opacities, larger on the left side, could be due to atelectasis but superimposed infection cannot be excluded. If any, there is a small right pleural effusion. There is elevation of the right hemidiaphragm. There is mild vascular congestion.	*No Cardiomegaly* Atelectasis[u] Pneumonia[u] Edema[u] Airspace Opacity Pleural Effusion

Figure 1: Example of a report from the MIMIC-CXR dataset. This table is taken as is from the paper [11]

we were left to evaluate the quality of our labeler ourselves. By our interpretation, it seems the labeler has an accuracy of over 93% [8].

[8]The automated labeler and its evaluation was performed by Maxime Fournier, another intern.

2 Contexte

It is in this context of shortage of radiologist, and while listening to a presentation by the CheXpert team, that the administrator of the CIUSSS Centre Sud de l'ile de Montreal (CCSMTL), Mathieu Mailhot , therefore decided to launch a pilot project to introduce AI into the hospital center. In Quebec, there is currently no AI solution playing a role in the diagnosis of a patient. This project therefore represent a first in respect to bringing an AI assisted tool to help the diagnosis of a patient in Quebec.

The CCSMTL is a rather large organization. It covers 67 medical clinics and 300 000 persons. While the CCSMTL is mostly dedicated to providing medical care to the population, its mission also extend to covering research, as indicated by the CIUSSS name (Centre Intégré Universitaire). They therefore saw fit to bring this project to their establishment.

For this project, three students were recruited to work under the supervision of Jean-Francois Samson. The exact make-up of the team can be seen in the figure 2

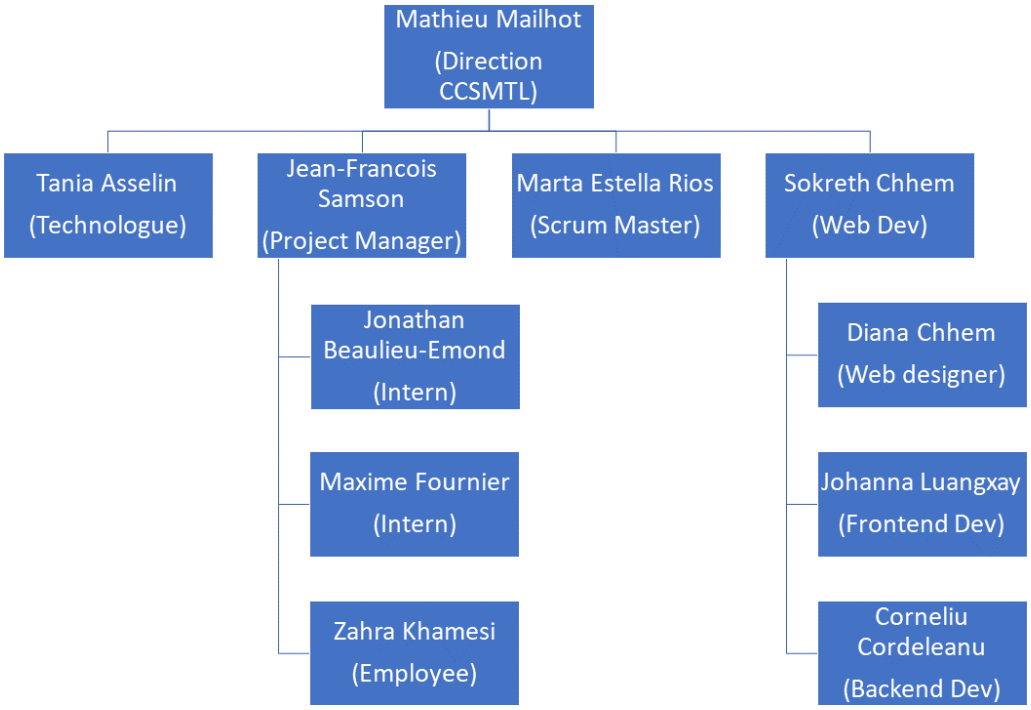


Figure 2: Organigram of the team working on the project as of December 2022. A technologue is responsible for the act of taking the x-rays of the patient.

We worked side-by-side with Tania Asselin and her team, who were in charge of gathering the raw data and anonymizing it. We later also had a web development team join us, to build the web application that would be used to display the results of the AI model.

The project is divided in 2 phases . The research phase, spanning 9 months, and the production phase, with the exact timing to be determined later.

3 Planning & Objectives

4 Objectives

As mentionned before, the project, which started as a classification one, ended up with the objective of building a weakly supervised object detection for 14 pathologies. To do this, we built a database of different publicly available datasets of CXR images, plus we used images from our local hospitals, and the annotations of the radiologist that reviewed those images. Our project therefore consists of 5 steps :

Figure 3: The 5 steps of our weakly supervised object detection model

These steps corresponds to 5 objectives :

1. Successfully converting text annotations to labels
2. Matching the different pathologies terminology to build a coherent database across datasets
3. Training a classification model to equal or superior level of a radiologist
4. Generating accurate enough pseudo label for the object detection to be used
5. Retraining a final detection model that will be fast and robust enough to be deployed

But before that, we will have to reproduce some of the results of the CheXpert competition, in order to validate our methodology before trying to improve it. This will also ensure that we are on the right track, by separating the errors that might arise from the model and its training than those that arise from the data.

However, the true objective is to have the radiologist used our tool at the end of this project. Sadly, the radiologist supposed to follow this project with us has been unable to participate, leaving

us with little feedback on our results. The specifics of the objectives are therefore subject to changes following future discussion with our team of radiologists, to see if they are satisfied with the performance.

1. Phase 1 : Development (May-December)
   (a) Review the litterature (May)
   (b) Reproducing the CheXpert's results (June-August)
   (c) Object Detection task (September-October)
   (d) Start working on the CCSMTL dataset (August-December)
   (e) Retrain using teacher-student techniques with pseudo-labels (November-December) ; outside the scope of the internship

---

2. Phase 2 : Deployment (January-March) ; outside the scope of the internship
   (a) To be determined

Delays in the delivery of the server (for computational ressoures) , and major delays in the delivery of the data explains why we had to spent so much time on litterature reviews and public dataset, as we were not able to start working on the CCSMTL's dataset until way later in the internship.

We were also only afforded one meeting with our radiologists, almost half-way through the internship, which changed our objectives and complicated the project further. Sadly, we were not able to obtain further meetings with the radiology team to receive help in curating our dataset.

5 Material

5.1 Hardware

To train our model (and potential future model), the CCSMTL has invested into its own server in the form of the dell poweredge r750, equipped with :

1. 2x Intel Xeon Gold 6338 2.00GHz
2. 16x 64gb DDR4 3200MHz (1024 Gb)
3. 4x Nvidia A100 80gb
4. 8x 1.6tb NVMe SSD (12.8 Tb)

5.2 Software

5.2.1 Experiment Tracking

To track our experience, we used Weight&Biases [12] .This tool allows us to track the different hyperparameters, the loss, the accuracy, and the model architecture. It also allows us to compare different runs, and to easily reproduce them. W&B also allows us to easily upload our model to their servers, and to download it from anywhere. This allows us to easily share our model with other people, and to easily reproduce our results. Finally, it allows us to easily run hyperparameters sweeps, which will allow us to find the best hyperparameters for our model once we finish our development.

5.2.2 Version Control

To keep track of our code, we used Git. This tool allows us to keep track of our code, and to easily share it with other people. It also allows us to easily revert to a previous version of our code, in case we make a mistake. Finally, it allows us to easily merge our code with other people's code, and to easily resolve conflicts. The code was hosted in Azure Devops, which also allowed me to link my pull request with unitary tests, and to automatically run them.

Azure Devops also come with an Agile board, which allowed me to keep track of my tasks and to organize my work.

While the server is more than powerful enough, it's important to mention its arrival was delayed , and it was not ready at the start of the internship.For almost the first two months of the internship, no computing ressources were therefore afforded to me and I had to use my own computer to debug my code , in the anticipation of the server's arrival. This is one of the factor that contributed to the delays of the project.

6 Deliverable

While I was mostly working alone on the programmation part [9], there was a few deliverable I had to follow. I also had to do a thorough documentation and explaination of the code as this project was to be continued by other interns in the future

[9] My supervisor was working on other projects, and the the other intern were focusing on other sub-project related to the same dataset

1. Litterature Review (First month ; while waiting for the server to arrive)
2. Monthly presentation with Powerpoint of the progress made
3. Reports for the attempts made to solve the problem (1-2 pages)
4. Bi-weekly Scrum meeting to discuss the details of the tasks at hands
5. Final Documentation ($\geq$20 pages)
6. README for the code ; with docstring for every main function

7 Data Analysis

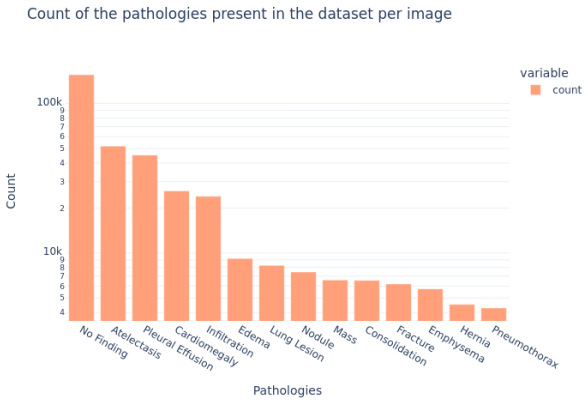
The first step of any ML project is to analyse the data available to us. Let's start with the CheXpert dataset. We will first look at the distribution of the classes. As we can see in 4(c), the dataset is not balanced. We will need to take this into account when training our model. We also can see that the three least represented classes are under 10 000 images, which might make it difficult to train a model that can accurately predict them. A good use of data augmentation techniques will probably be necessary to train a model that can accurately predict these classes.

Now looking at the distribution of the classes in the validation set, we can see that the classes are more evenly split. This is a good sign, as it means that the validation set will be a good representation of the performance on the different classes. This is however not true for all classes, as Fracture has no images at all present in the validation dataset. Four other classes also have less than 10 images present, which will make it difficult to know for sure whether the model is able to predict them accurately. The results for those classes will most likely need to be ignored.

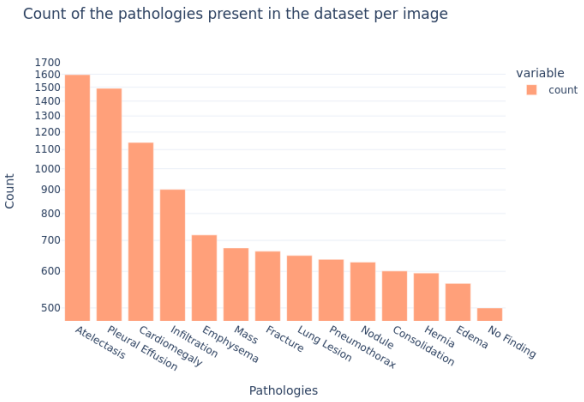
For the CCSMTL dataset, we can see in the training dataset 4 that we have an even stronger imbalance than in the CheXpert dataset. Images without any disease constitues almost 25% of the dataset, with over 100 000 images while some classes have less that 5000 images. We will have to correct for this while training, with strategies such as weighting the loss, data augmentations and undersampling the classes with too many images.

In the validation dataset, we did not simply randomly sampled the images, we can see the distribution of images differs from the training dataset. While there is still an imbalance in between the different classes, we can see that we still have a minimum of 300 images per classes, which should be enough to evaluate the performance of the model. We should however stay weary of this imbalance, since if we simply used a loss (such as the cross entropy), to see the performance of our model, the overrepresented classes might also be overrepresented in the loss, and thus bias our choice of hyperparameters. To correct for this, we can use metrics that takes into account the imbalance present in our dataset, such as the F1-score [10], or the AUC. [14].

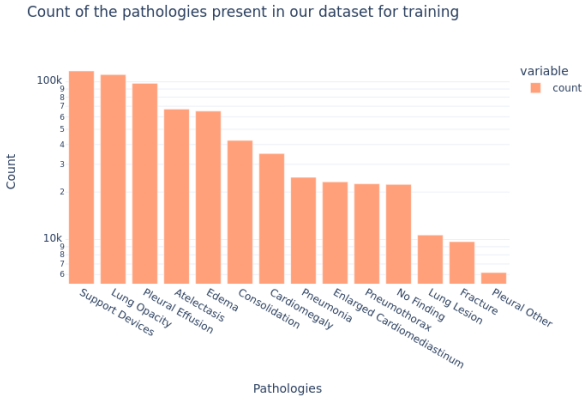
[10] In retrospect, since the F1-score is not a smooth function, it is an imperfect function to keep track of our model's improvement. While it's a useful metric to evaluate the performance, the probabilistic F1-score [13] could have been used instead.



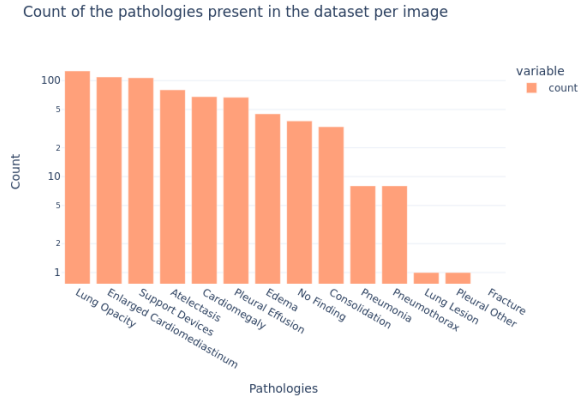
((a)) CCSMTL's training dataset



((b)) CCSMTL's validation dataset

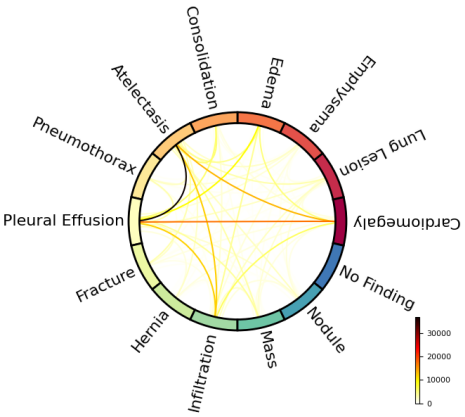


((c)) CheXpert's training dataset

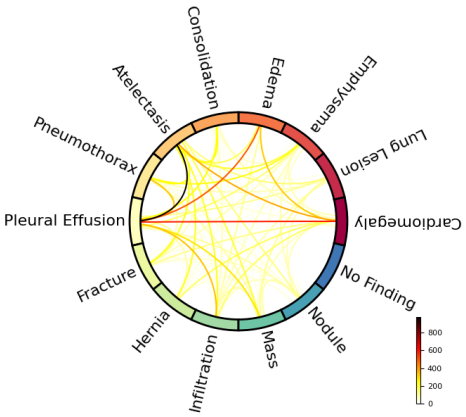


((d)) CheXpert's validation dataset

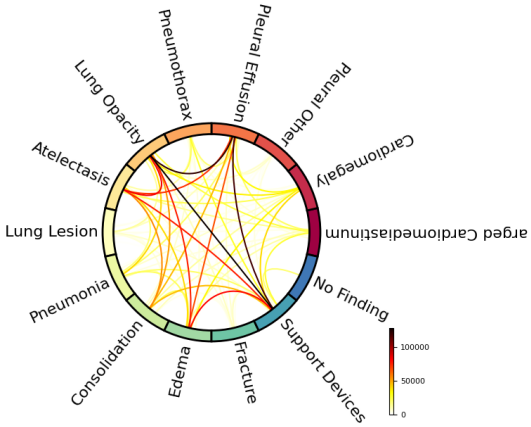
Figure 4: Histogram for the different labels present in the CCSMTL and CheXpert dataset



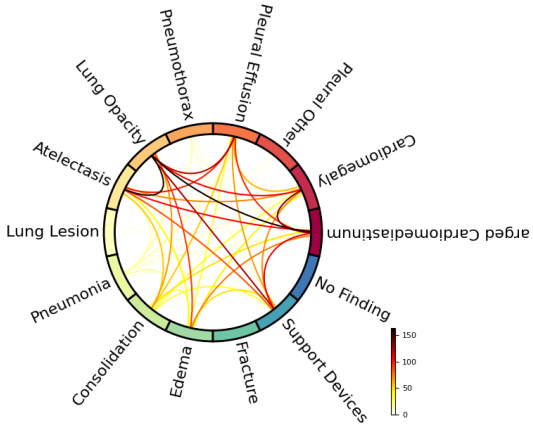
((a)) CCSMTL's training dataset



((b)) CCSMTL's validation dataset

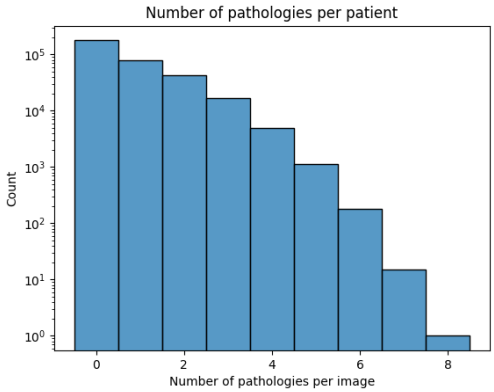


((c)) CheXpert's training dataset

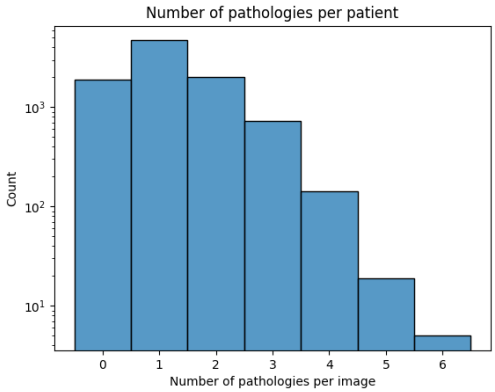


((d)) CheXpert's validation dataset

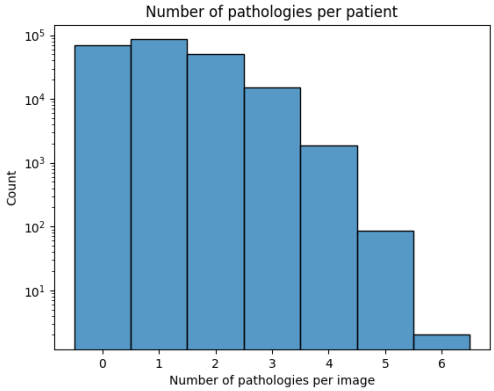
Figure 5: Histogram for the different labels present in the CCSMTL and CheXpert dataset



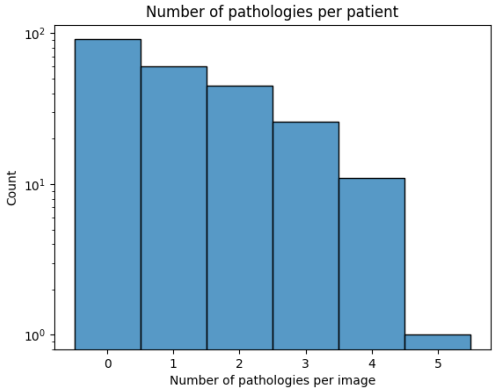
((a)) CIUSSS's training dataset



((b)) CCSMTL's validation dataset

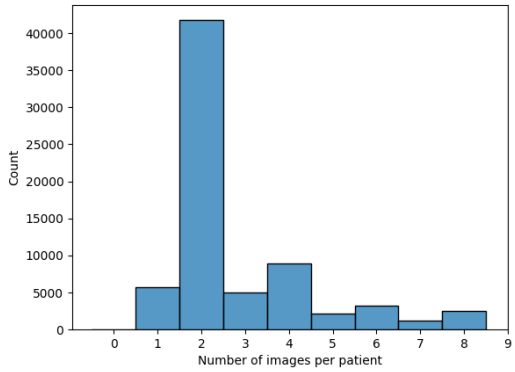


((c)) CheXpert's training dataset

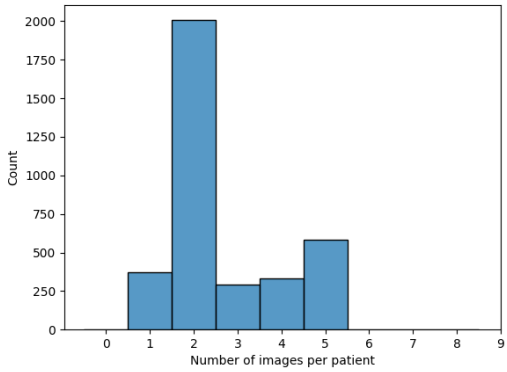


((d)) CheXpert's validation dataset

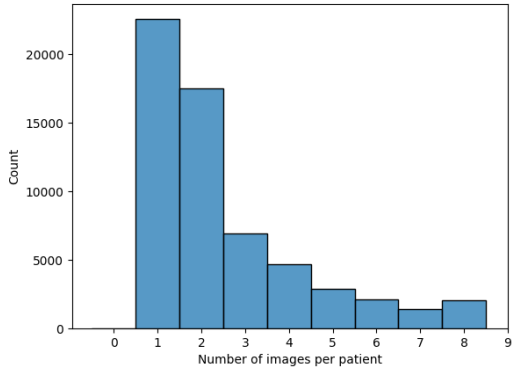
Figure 6: Histogram for the different labels present in the CCSMTL and CheXpert dataset



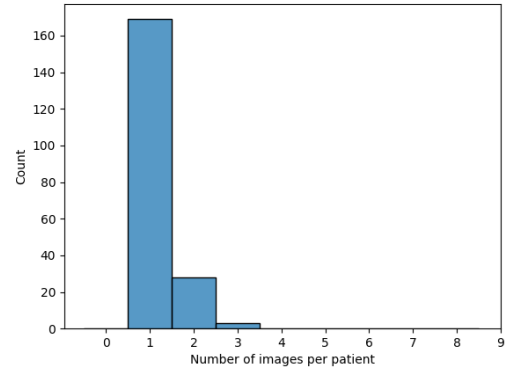
((a)) CCSMTL's training dataset



((b)) CCSMTL's validation dataset



((c)) CheXpert's training dataset



((d)) CheXpert's validation dataset

Figure 7: Histogram for the number of images per patient present in the CCSMTL and CheXpert dataset

Another important thing to look at could be the correlation between diseases, to identify potential biases in the data. To account for uncertainty labels, we will simply replace them by a positive label for this analysis.

Looking at the Chexpert's correlation graph (see 5(c) and 5(d)) , we can see that the correlation between diseases is quite high. This is slightly worrying, as it means that the model could be biased, simply learning this correlation instead of recognizing one of the classes. We can also see that the correlation is higher in the validation dataset than in the training dataset. This is expected, as the validation dataset is smaller and therefore more likely to have a higher correlation between diseases.

However, when we look at the CCSMTL's correlation graph (see 5(a) and 5(b)), we observe a way lower correlation level between pathologies. It might also be due to a difference in the population but this explanation does not seem quite likely

Assuming the preponderance of diseases is the same between the population visiting the Stanford's hospital and the CCSMTL, we are left to assume that the quality of the data is different from the Stanford dataset. Whether it is better or worst is to verify.

As mentionned earlier, the data is unbalanced! While typically , we would simply undersample

Classes	Uncertainty ratio	
	CheXpert	CCSMTL
Enlarged Cardiomediastinum	115%	N/A
Cardiomegaly	30%	5%
Pleural Effusion	13%	29%
Pneumothorax	16%	20%
Lung Opacity	5%	N/A
Atelectasis	101%	49%
Pneumonia	311%	N/A
Consolidation	188%	34%
Edema	25%	224%
No Finding	0%	0%
Fracture	N/A	17%
Hernia	N/A	17%
Infiltration	N/A	33%
Mass	N/A	27%
Nodule	N/A	31%

Table 2: Uncertainty ratio for each class

the overrepresented classes or the opposite, oversample the under-represented classes, it is not an option here. Looking closely at the data(see 6(a) 6(b) 6(c) 6(d)), we can see our problem is not a simple classification, but a multilabel classification tasks where more than one class might be present in the input. We also see that we have a lot of images with no pathology at all in the CCSMTL dataset. This is a problem as it means that the model will be biased toward answering no pathology are present. We therefore need to undersample this category!

To resolve the imbalancy, we can also assign each positive example of a class with a weight, therefore increasing the importance of the positive examples of the under-represented classes. This is done by using the class weights parameter of the loss function. This is a good solution, but it is not perfect of course as one class might have a higher preponderance in the loss function.

Looking at the repartition of certain vs uncertain label, we found that there was an enormous discrepancy between the classes when it came to the ratio of uncertain vs certain labels. As can be seen in the table 2, certain class can have up to thee more time uncertain label than certain. This confirms a lot of our hypothesis about the amount of noise present within the dataset. We will have to see if this noise stop the model from learning, or simply act as a regularization technique. Another worry is the difference between the CCSMTL's data and the CheXpert data. We can see we have a quite different ratio of uncertainties. This could mean I have wrongly label our data. We will have to see if this is the case later on.

Secondly, our data does not come in the form of a simple image. The input is composed of multiple images present in the patient file for a specific exam, typically between 1 and 8 images 7(a), with most patient's exam having two images or less. While a RNN could be use to deal with a sequence of image, no such model exist for medical imaging. Instead, we will use a CNN to process each image independently, and combine the output with a simple addition before applying the sigmoid activation function.

To further simplify the model, we also explored two other option

7.1 Option 1 : Only use 1 frontal image

Our first attempt was to use only one frontal image per patient. This is a simple solution, but it is not optimal as it does not take into account the other images present in the patient file. However, we quickly realize that this was not a good idea as the model was underperforming with some specific pathologies. This was most likely due to these pathologies having been identified through the other images present in the patient file, and especially the lateral images. Further investigation confirmed our hypothesis, as some pathologies like pleural effusion were mostly confirmed by the lateral images.

7.2 Option 2 : Use two different set of weights for the feature extraction of frontal vs lateral images

To ensure the model learned in an optimal manner from both lateral and frontal images, we tried to use two different set of weights for the feature extraction of frontal vs lateral images. This however effectively double the number of parameters to train, and led to a significant increase in the model capacity and overfitting. This option was therefore also rejected.

Finally , we opted to use two of the images present in the patient file, and to combine the output of the CNN with a simple addition before applying the sigmoid activation function. This is a simple solution, but it is not optimal as it does not take into account the other images present in the patient file. It however will help simplify training as the model will almost always have the same number of input images (2, more rarely 1), and will also help with the generalization of the model as it will be trained on a wider variety of images.

To undersample the empty images, we will ponder them with a weight of 0.1, and the other images with a weight of 1. This will effectively leave us with a dataset containing the equivalent of only about 20 000 empty images.

8 Preparing the data

Before sending our data into a machine learning algorithm , we need to think about the preprocessing that we can apply on it to facilitate the learning process. For our use case, with deep learning models, it is typical to split this step in two part : augmentations and normalization.

For data augmentations, we used a combination of random horizontal flipping of the image, as the lungs are symmetrical,color jittering, to add random noise in the images, affine transformation (see [11]), and a mix of grid distortion and Elastic transformation to simulate a greater variance in the shape of the specified pathologies. All the augmentations were done with Albumentations, as to avoid making mistakes by writing our own[15]. The detail of the augmentations :

- **Affine** : randomly apply affine transformations : translation, rotation, shear and scale with probobility p_0. The translation is done with a maximum of 10% of the image's width and height, the rotation is done with a maximum of 15 degrees .
- **ColorJitter** : randomly change the brightness, contrast and saturation of the image with probability p_1, and a maximum of 0.1 for each of the three parameters.[12]
- **HorizontalFlip** : randomly flip the image horizontally with probability p_2

[11] Albumentations's documentation

[12] While it may sound strange to apply color jittering on a grayscale image, it is still possible to apply it, as it will simply affect the contrast, brightness and saturation of the image.

- **GridDistortion** : randomly apply grid distortion with probability p_3, and the following parameters : num_steps=5, distort_limit=0.2, interpolation=1, border_mode=0, value=None, mask_value=None
- **ElasticTransform** : randomly apply elastic transformation with probability p_4, and parameters : alpha=0.1,sigma=20 and alpha_affine=40

Data augmentation such as this serves two purposes. The first one is to allow the model to generalize better, as it is given a wider variety of examples (albeit partially synthetic examples) . The second purpose is to avoid overfitting as the model will have a harder time simply memorizing the examples instead of actually learning the desired characteristic within the image.

Too much data augmentations can also be a problem, as it can stop the model from learning the desired characteristic by applying too much noise to the image.

The second step, normalization, serves to assure that the distribution of the image stay within a specified norm. It is typical to bring the images values between 0 and 1, and , if using a pretrained model as it is our case, to normalize the data according to the specified mean and standard deviation value of the images used to pretrain the model (in our case ImageNet).

It is also possible to use histogram equalization. This method will normalize image with respect to local values instead of the global maximum and minimum. Improving on this idea we used the Contrast Limited Adaptive Histogram Equalization, an often used method in medical imaging [16] [13], to further improve the normalization and contrast of our images. I hope this will help with the cross-generalization of our model on the different datasets.

8.1 Dealing with uncertainty labels

Since the labels are derived from the radiologist's report, they often include uncertainties express with vocabulary such as "might" , "maybe" , "possible" , etc. This is a problem as it is not clear what the label should be. therefore decided to assign a flag of -1 to all uncertain label (just as it was done with CheXpert). Following this paper [9], which tested different ways of dealing with those uncertainty, we replaced uncertain label with a random label sampled uniformly between 0.5 and 0.85, as to imply the label is more likely present than absent if the radiologist mentionned it as a possibility.

This will also act as a form of regularization, as a form of label smoothing [14], teaching the model to be less overconfident about its prediction. By default, the cross entropy loss will try to minimize the distance between the predicted label and the true label. This is a problem as it will put a pressure on the model to simply answer 0 or 1, without leaving much room for uncertainty. If we wish to interpret the output of the model as a probability, we need to ensure the model is not overconfident about its prediction. Label smoothing will help with this, as it will teach the model to be less confident about its prediction.

This will help when communicating with the radiologists, as they will naturally try to interpret the output of the model as a probability. This will also help with the interpretability of the model, as it will be easier to explain to the radiologists why the model was wrong, as it should be less confident about its prediction.

It is important to note that such uncertain label should not and are not in the validation set as to avoid having validation results that include a lot of noise, which would take away a lot of our trust in the results.

Just as with the validation set, the test set should not contain any uncertainty either.

[13] see towards data science

[14] towards data science

9 Model architecture

9.1 Model 1 : DenseNet

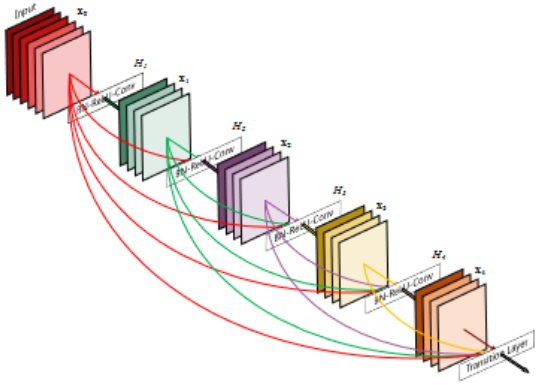


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Figure 8: Figure 1 from the densenet paper [17]. It shows how the layers are all interconnected

The densenet model is a CNN model that is composed of dense blocks. Each dense block is composed of a series of convolutional layers, each of which is connected to all the previous layers in the block (see 8). This allows the model to learn features at different scales, and to combine them in a more efficient manner. The model is also composed of a transition layer, which is used to reduce the number of feature maps and to control the growth of the model. The model is also composed of a classifier, which is composed of a global average pooling layer and a fully connected layer.

This model released in 2018 improved on ResNet with its dense blocks, and was the state of the art for a while. The stanford team working on ChexNet found it was the best model available to achieve the highest AUC on CXR images [2].

9.2 Model 2 : ConvNeXt

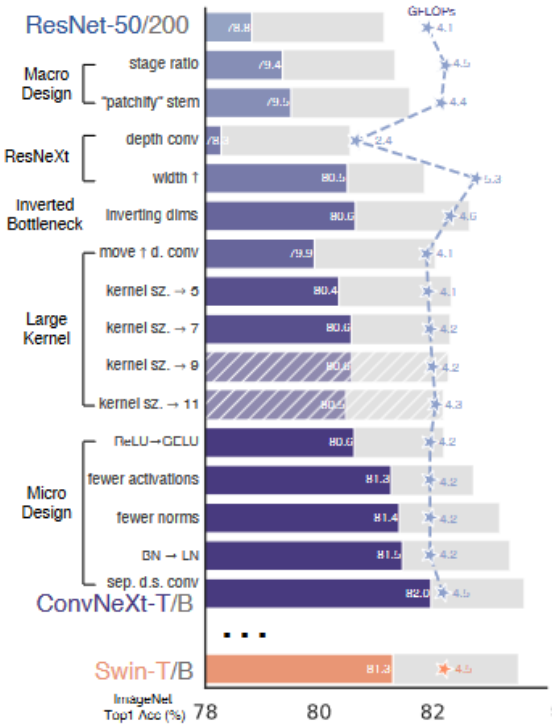


Figure 9: Figure 2 from the ConvNeXt paper [18]. It details the different improvement made to the resnet architecture and the gain in performance obtained following each modification.

The ConvNeXt model is another CNN model, this time released in 2020 and improving on the default Resnet architecture once again. They gradually implemented many of the features of vision transformers in order to "modernize" the typical CNN architecture. In doing so, they created an architecture which outperformed many SOTA models on the ImageNet competititon.

The details of these modifications can be seen in the figure 9, which is taken from their paper. It replaced densenet as the "default" convolutionnal model

9.3 Model 3 : EfficientNet and YOLO

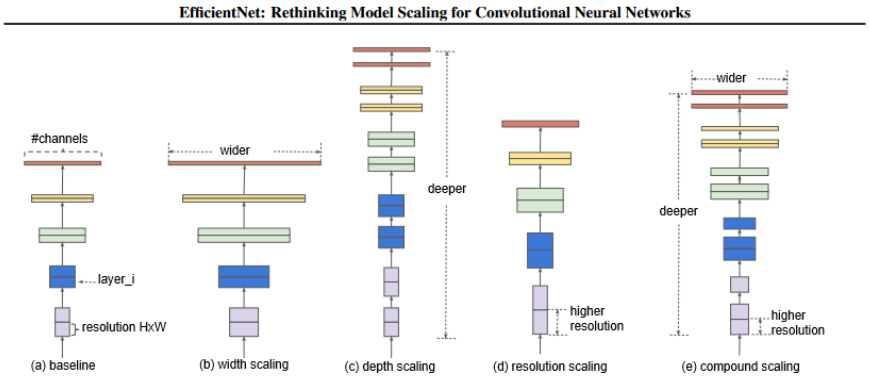


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Figure 10: Figure 2 from the ConvNeXt paper [19]. It shows the different approach that deep learning model can take to model scaling, compared to their version.

EfficientNet was another model improving on the basic building blocks of CNN, in order to improve the efficiency (see 10). While it is not the best one, this approach has led to further development. Such improvement came in the form of the You Only Look Once (YOLO) [20] object detection model, which quickly became a SOTA model. Further improvement have allowed this model to keep its position as one of the best pretrained model publicly available.

9.4 Model 4 : Deit & Transformers

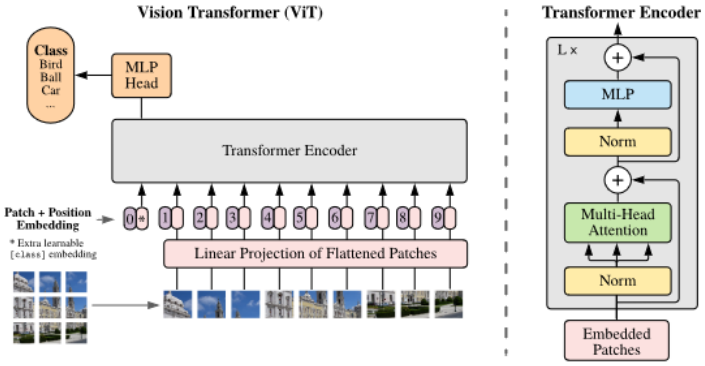


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Figure 11: Figure 1 from the An image is worth 16x16 words paper [21]. It shows the general approach of an image transformer.

Transformers [21] are a new revolution in deep learning. First developped to deal with natural language processing (NLP), they were later introduced in image models, successfully improving on CNN's. While CNN still keep a few key elements on their side(they usually perform better to extract local information in an image), transformers split the image in blocks, treating them as different element of a sequence (see 11). Among other things, this allows them to better extract global feature present in an image.

Later, the facebook research team improved on this general idea by pretraining the model on a similar, but different task, a now common approach when using transformers [22]. This allowed them to achieve a greater accuracy in the ImageNet competiton than previous models.

10 Training

10.1 Resolution

While our original images have a very high resolution (up to 4096x4096), we decided to downscale them to 384x384 pixels. While some information might be lost due to this downscaling, having higher resolution images can lead to more overfitting, and it is also more computationally expensive to train on such high resolution images. An added benefit of this is that it will allow us to train on a larger batch size, which will allow for better statistics in the batch/layer normalizations, and will also allow us to train faster.

It also has already been demonstrated in this paper [23] that , from a range of 224x224 to 1024x1024, the performance of the model does not change much.

10.2 Loss function

To train a classification model with a multilabel output, we use the Binary Cross Entropy. We therefore treat each class as a binary classification problem.

To account for our dataset imbalance, we will also use weights for the positive example of a class (P_c), and weights for the specific classes (w_c) such that the loss will be

$$L = \frac{1}{N}\sum_{n=0}^{N} -w_n[p_c y_n log(\sigma(x_n) + (1-y_n)log(1-\sigma(x_n)))] \tag{1}$$

10.3 Learning Rate Scheduler

II will use a learning rate scheduler to help the model converge faster. We will use a OneCycle learning rate [24], which will decrease the learning rate as the training progresses. This should help the model converge better, as it will first allow it to explore a more vast parameter space, before allowing it to converge into a local minimum.

10.4 Early stopping

Early stopping is a technique used to avoid overfitting. It consists of stopping the training when the validation loss starts to increase. I will implement a patience mechanism so that the training will stop only if the validation loss has not improved for a certain number of epochs.

10.5 Optimizer

Multiple optimizer exists, building on one another . While none is perfect, or definitely better than another, AdamW [25] [15] is oftenly use as it tends to converge rapidly, thus allowing for far smaller training time.

AdamW is a slightly modified version of the optimizer Adam (which itself is a modified version of AdaGrad and RMSProp), and fixed an error in the L2 regularization from the original implementation.

AdamW is also chosen as it has demonstrated its abilities to converge easily with the default hyperparameters. This also allows to diminish the needs for finicky hyperparameters tuning.

10.6 Metrics

In any unbalanced dataset, the accuracy won't be enough to determine the performance of a classifier. We therefore decided to look into the Area Under the Roc Curve (AUC-ROC) [16] and the F1-score[17].

The F1-score is a metric which combines the precision and the recall, simply written :

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2}$$

While the F1-score does allow to evaluate the performance of our classifier, it requires to first assign a binary class to our model's output. This in turn require to preset a threshold, which, without the results first, will usually be arbitrarily chosen to be 0.5. While this is not ideal, as this arbitrary threshold might hinder our model's performance, it is still a good metric to use to evaluate how our model would be received by radiologists.

On the other hand , the AUC-ROC is a metric which does not require to preset a threshold. It is simply the area under the curve of the ROC curve. However, to have a perfect AUC-ROC, our model would have to be overly confident in its predictions, always predicting probabilities very close to 0 or 1. While the prediction might be accurate, it makes it harder to explain to a radiologist why the model predicted a certain class over another. This is why we will use both metrics to evaluate our model's performance.

10.7 Classification Head

While we use a pretrained model for our feature extraction, leaving us little to change on that side, the classification head is usually a simple fully connected layer, allowing us some leniency if we were to want to change it. However, such modification are usually not required for a simple classification model . However , as we switch from classification to object detection, we might want to change this later.

For now, we might mostly just want to work with the dropout parameter. DropOut is a useful technique to avoid overfitting and increase model generalization [18]

[15] See AdamW' documentation
[16] towards data science
[17] towards data science
[18] towards Data Science

10.8 Tips and Tricks

While dealing with a project of this size, every optimization counts to avoid training a model for days. Here are some tips and tricks I used to make sure my code ran at the fastest possible speed.[19]

1. label-smoothing : A common use technique of adding or removing a small value (e.g 0.1) from the label in order to smooth out the gradient (e.g $0 \rightarrow 0.1$, $1 \rightarrow 0.9$)
2. Clip-Norm : The weight of the model might sometimes grow larger and larger leading to overflow and infinite/Nan values! To avoid this, the gradient's norm might be clipped at a maximum value (e.g 1). This also acts as a form of regularization
3. Timm : The code use the timm library [27] in order to load a model, allowing for an easy use of a multitude of pretrained model.
4. Input Channels : While models are pretrained to use 3 channels (RGB), we can use a single channel (e.g grayscale). This uses only one of the convolution kernel at the first layer of the model, and thus reduce the number of parameters to train. However, it might require more training as it will affect the input of all the other layers of the model.

11 Preliminary Results

All these results for CheXpert can be found at Weight and Biases. Please note you need to be granted access to the project in order to see the results.

11.1 Reproducing CheXpert baseline

While we do not have access to the hidden test set from CheXpert, we still have access to their validation dataset. However, as can be seen in 4(d), three classes are not available in the validation dataset. We will therefore have to limit our comparison to the classes present in the validation dataset , and to keep in mind our results will most likely be higher than they would be on the test set. However, to keep things simple, we will not proceed to model ensembling as this will simply put a high computational cost for a simple verification step.

I will also use positive weight to account for the imbalance of the dataset. The positive weights were calculated using the following formula : $pos\_weight = \frac{N_{neg}}{N_{pos}}$ where N_{neg} is the number of negative samples, and N_{pos} is the number of positive samples for a given class. Their value range in between 1 and 95. The values are reported in the table 3. I suspect that high values might scramble the loss for the batches containing the rare positive samples of that class. I might later want to cap the maximum possible value of the positive weight to avoid this.

While we use a weighted sampler for our dataset,for CheXpert, all the training examples were sampled with equal weights. The sampler only allowed to randomly select 50 000 images per epoch.[20]

[19] Note that this is not a complete list, and that there are many other ways to optimize your code. I was very inspired by this article [26]

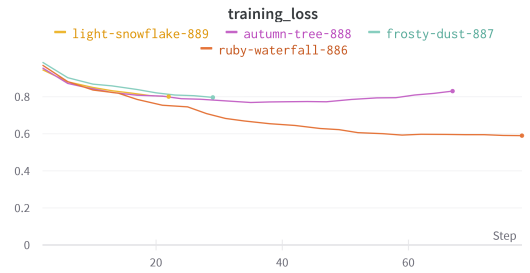
[20] While a typical strategies for classification problem would be to sample every class equally, it's a bit more trickier in our case since images can often contain more than one class. Sampling 20 images of the class A will therefore impact the number of images of class B present. This is why we switched to using a weighted loss instead.

Classes	Weights
Opacity	1.97
Air	16.76
Liquid	1.16
Cardiomegaly	12.15
Pleural Other	94.76
Pleural Effusion	3.07
Pneumothorax	21.06
Lung Opacity	1.68
Atelectasis	9.66
Lung Lesion	27.71
Pneumonia	50.26
Consolidation	33.28
Edema	6.11
Fracture	22.77
No Finding	6.58

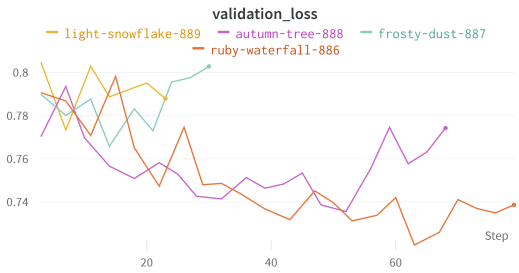
Table 3: Positive weights used for the loss , from the CheXpert dataset

11.1.1 First Attempt

Using everything we describe so far, we obtain results as shown in 4. While the model did learned a bit from the data, we are underperforming the baseline by a large margin. I will therefore have to try to improve our model.



((a)) Training loss



((b)) Validation loss

Figure 12: Training curves for our first attempt at reproducing the CheXpert baseline

Our second attempt will be at removing the positive weights. Currently, we are only using the weight in the training loss. However, this multiplication factor could hide a potential overfit on the training set. Currently, however, it seems the opposite. The training loss plateau at around 0.6, while the validation loss varies between 0.8 and 0.6. This would usually indicate that the model is underfitting and need to have a bigger capacity.

Now looking at the f1-score and AUC 4, we can compare the performance of our model to the baseline. We can see that we are underperforming

While the performance for the lung lesion and the pneumonia are not good, we can remember from figure 4(d) that these classes are barely present in the validation dataset (lung lesion only

Classes	AUC CheXpert	F1-Score (Radiologist's estimation)	AUC (Ours)	F1-Score (Ours)
Cardiomegaly	0.832	0.678	0.74	0.62
Pleural Effusion	0.934	0.737	0.90	0.74
Pneumothorax	N/A	N/A	0.87	0.23
Lung Opacity	N/A	N/A	0.88	0.85
Atelectasis	0.858	0.692	0.71	0.63
Lung Lesion	N/A	N/A	0.26	0
Pneumonia	N/A	N/A	0.70	0.11
Consolidation	0.899	0.385	0.85	0.41
Edema	0.941	0.583	0.86	0.53
No Finding	N/A	N/A	0.84	0.93

Table 4: Performance of our first attempt at reproducing the CheXpert baseline. The estimation of the radiologist is the one provided by [1]

has one image). Ingoring those, we find that, while comparing with the AUC, the model has underperformed, we did achieved interesting f1-score. We find that our model seems to perform similarly to a radiologist's on the classes we do have comparisons. This is a good sign, as it means that our model is learning something from the data.

While our results are slightly lower than the values reported by the CheXpert team, we however achieve close enough value to be confident in our implementation. However, only 5 classes were reported in the paper, and we have 10 classes. We will therefore have to compare our results with other sources.

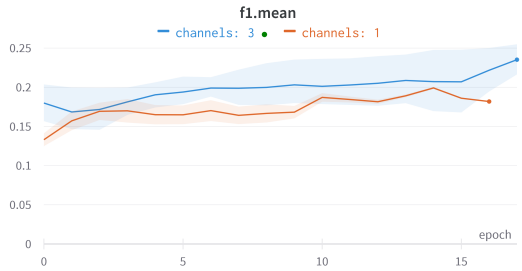
If you paid attention up to this point, you will also have notice that CheXpert had 14 classes, and not exactly the same as we have! This is due to a talk with our radiologist, where they identified some classes as being too similar to be useful, and we thus removed them.

Rerunning the training without the positive weights, we can now more clearly see that the model is clearly overfitting. We should therefore try to switch to model with smaller capacity, and to increase the data augmentation.

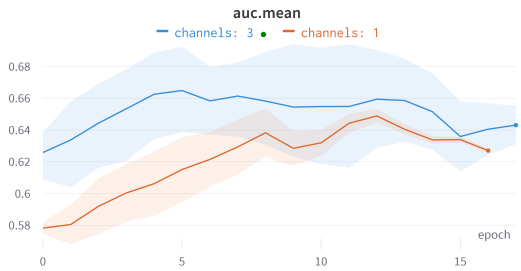
11.1.2 Second Attempt

Without the positive weight, as suspected, we can clearly see the overfit of the model 15(a) . Our AUC does increase but at the price of a diminished F1-score. While it helps avoid false positive, it also makes it harder to identify the true positive.

Another interesting thing to remember is that the distribution of positive and negative samples between the training and validation dataset is not the same! So while the model can achieve a low loss by simply providing negative answer at first, it does not mean that it will be able to generalize to the validation dataset. This is one of the reason why we probably should continue to use positive weights.

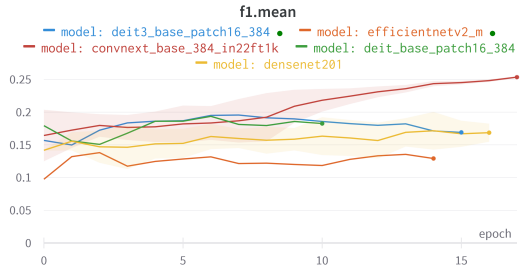


((a)) Average f1-score over all classes

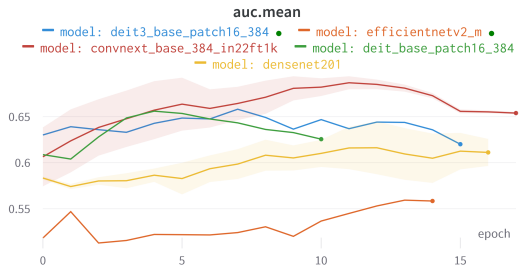


((b)) Average AUC over all classes

Figure 13: Comparisons of the experiments results grouped by number of channels of the input images for the average AUC and f1-score. The curve are smoothed out to help the analysis, and the error bars are the min-max of the values. The runs were grouped and averaged, with the min-max values being displayed as the range

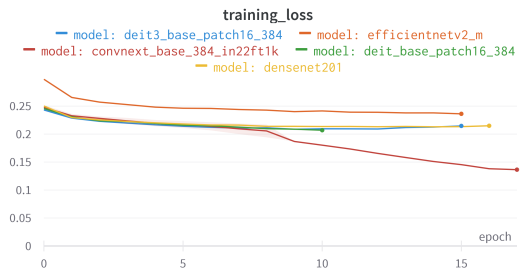


((a)) Training loss

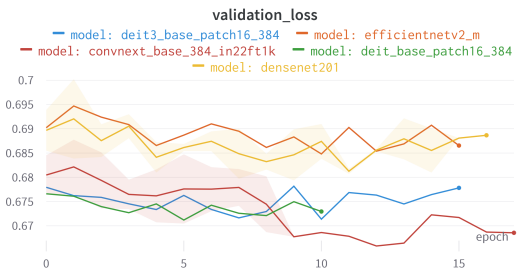


((b)) Validation loss

Figure 14: Comparisons of the experiments results grouped by model.The curve are smoothed out to help the analysis, and the error bars are the min-max of the values.The runs were grouped and averaged, with the min-max values being displayed as the range



((a)) Training loss



((b)) Validation loss

Figure 15: Training curves for oursecond attempt at reproducing the CheXpert baseline. The different runs are grouped by models.The runs were grouped and averaged, with the min-max values being displayed as the range

I can now separate our results by the model choice to see which one is the best.As can be seen in the figure 15(a) and 14(b), while the DenseNet model does learn from the data, it still underfit while the convnext model is able to overfit our training data. This does seem to indicate that we need a more complex model than the densenet. However, the convnext model is able to achieve a better AUC than Densenet. It means that while by itself, the training curves seem to indicate an overfitting, since our f1 is still increasing, our model is learning to classify better. This means the difference in loss in between the training and validation set is seemingly due to the difference in distribution of the two.

For our next experiment, we should therefore continue with the ConvNeXt model, and increase the data augmentation. Depending on those results, we can later try to add different data augmentations technique or a larger version of the ConvNeXt model.

I can also consider to use the positive weight depending on whether we prefer optimizing the AUC or the F1-score. Future discussions with the radiologist that will be using our model will help us decide which metric is the most important. In the meantime, following direction from my supervisor, I will try to optimize the F1-score as it is the metric that will allow ourselves to compare the performance of the model to the radiologists. It will also be more intuitive to have a model that has a threshold at 0.5, and that express some uncertainty when the probability is between 0.4 and 0.6 [21]. This is not the case with the AUC, as it is a metric that gains points for being very close to 0 or 1. I still personally think the AUC might provide a better evaluation of our model, but I will continue under this new directive.

Finally, we can see that while the validation loss is a bit chaotic, the training loss decrease while the f1-score is still increasing. This indicate that the model is still learning from the data. I should therefore continue to train the model for a few more epochs. To do that, we should therefore increase the patience to 10 or 20 epochs.

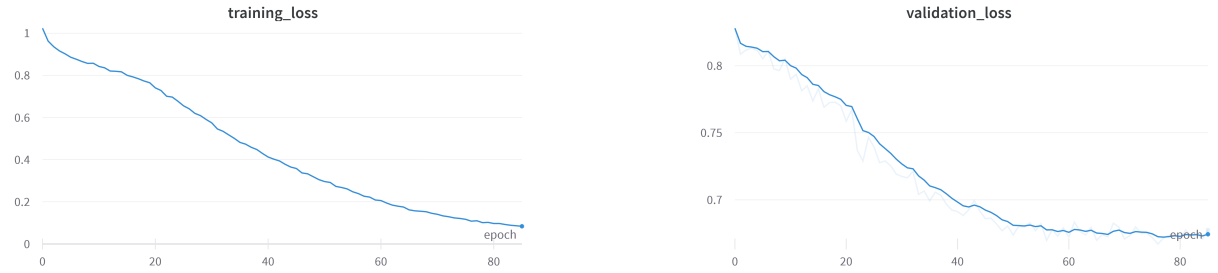
Just as we compared the performance by the choice of model, we can do the same to compare the performance of using 1 vs 3 channels. Doing so, we can observe in the figure 13(b) that the model with 3 channels is able to achieve a better AUC and F1-score. This is most likely due to the fact that the model is able to extract more information as we did not simply reapeated the same information on the three channel, but instead concatenated the min-max normalized image,

[21] However there exists some rescaling formula to compensate for this. We would usually set our optimal threshold based on the validation results before applying our model to the test dataset, but i do not have access to it, and the validation set is already too small to be split in two.

and two image treated with the CLAHE algorithm, albeit with two different threshold (2 and 4). But while it performs slightly better, it is not by a large margin, but it does increase the training time significantly by putting a heavier load on the CPU. This is why we will continue to use the 1 channel model for the rest of the experiments, until we reach our final results.

11.1.3 Third Attempt

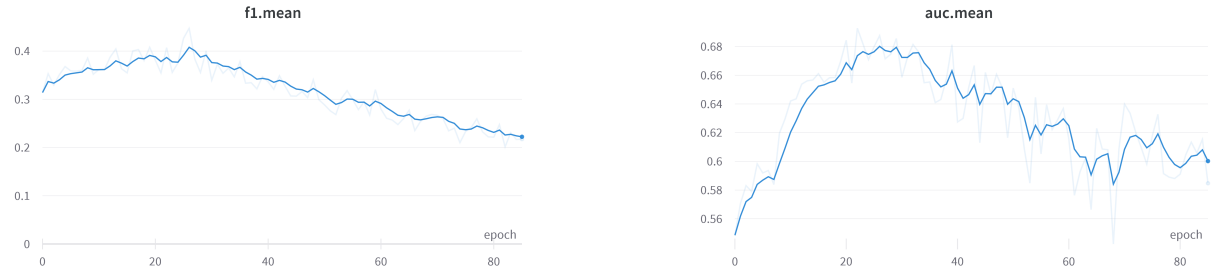
Applying the lesson learned, I tested the convnext base model with the 1 channels, and with/without the positive weights. I also increased the patience to 20 epochs. The results are shown in the figure 16(a) and 17(a). We can see that the model with the positive weights is able to achieve a better AUC and F1-score. All the details of this run can be find in Weight and Biases (please note that this links will only work if you were invited to the project)



((a)) Training loss

((b)) Validation loss

Figure 16: Training curves for our third attempt at reproducing the CheXpert baseline



((a)) Training loss

((b)) Validation loss

Figure 17: Metrics curves for our third attempt at reproducing the CheXpert baseline. The average values include classes with no positive labels in the validation set. It should therefore only be used qualitatively. The curves are smoothed out for an easier visualization.

Classes	AUC CheXpert	F1-Score (Radiologist's estimation)	AUC (Ours)	F1-Score (Ours)
Cardiomegaly	0.832	0.7	0.74	0.60
Pleural Effusion	0.934	0.7	0.90	0.65
Pneumothorax	N/A	N/A	0.87	0.22
Lung Opacity	N/A	N/A	0.88	0.834
Atelectasis	0.858	0.68	0.67	0.47
Lung Lesion	N/A	N/A	0.26	0.67
Pneumonia	N/A	N/A	0.70	0
Consolidation	0.899	0.4	0.75	0.45
Edema	0.941	0.6	0.86	0.55
No Finding	N/A	N/A	0.89	0.93

To obtain these results, we used the following hyper-parameters :

- backbone : convnext large
- batch size : 64
- learning rate : 1e-4
- weight decay : 1e-2
- dropout : 0
- num samples per epoch : 50 000
- optimizer : AdamW
- scheduler : OneCycle
- patience : 20
- label smoothing : 0
- clip norm : 1
- epoch : 200
- channel : 3
- augment prob [1, 1, 0.5, 1, 1]

11.1.4 Lesson learned

The results of our last run was a bit of a deception. While we did manage to achieve a lower BCE loss than before, we did not manage to improve the AUC and F1-score significantly. While we cannot know for certain , a few factor more likely contributed to this. The first one is the three classes with low representation in the dataset. Since only randomly predicting a few true positive greatly increase the average AUC and F1-score without improving really the model's ability to predict the true positive, it biases our choice of model towards that one epoch which performed better in these unrepresentative categories. In future runs, we should either remove these classes from our metrics calculation or from our model's prediction. This is why from now we will exclude fractures and lung lesions as classes for our model.

We also see that the Cross-Entropy does not allow for a good comparison between the different models,as it does not always correlate well with the AUC and F1-score. Ideally, a loss that allow to optimize more closely the AUC and F1-score would be better suited for this task. I will try and explore if such solution exists. Since our model's loss does not correlate well with the AUC and F1-score, we should use the F1-score as the metric to determine when to stop training. Our patience will therefore from now on be based on our F1-score instead of the loss.

Finally, lowering the learning rate, in combination with the usual OneCycle scheduler allowed the model to reach a lower training and validation loss.

11.2 Reproducing the CheXpert competition's results

The CheXpert dataset was released as part of a competition, where the goal was to achieve the best possible performance on the hidden test set. The competition was won by a team from Stanford, who achieved a score of 0.93 on the hidden test set. The results of the competition are reported in the table 1. The metric used was the ROC-AUC. I will therefore use the same metric to compare our results with theirs. However , we also implemented the F1-score (the harmonic mean of the precision and recall), which will make it easier to compare with the radiologist's performance in order to convince them to use our model.

11.2.1 DeepAUC

Attempting to reproduce the results of DeepAUC [10] proved not to be possible. The snippet of code , provided by the author, did not reproduce the results as expected. No pretrained model was also given. Attempt to use their loss function with our dataset also proved to be unsuccessful. I therefore decided to not include their results in our comparison, and to not pursue this avenue.

11.2.2 Hierarchical Learning

The hierarchical learning method [9] is a method that uses a hierarchy in between the different classes to more accurately predict them. Thanks to bayesian probabilities, we know that

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

If we use the parent class as B, with the child class as A, we can therefore redefine the probability of the child class such that

$$P(Child|parent) = \frac{P(Child \cap parent)}{P(parent)} \approx P(parent|Child)P(Child) \tag{4}$$

I could use P(Parent) as the prior probability of the parent class, as defined by the preponderance in the dataset. However, as I base myself of [9], I observe they did no such thing and I decided to not do it either. While this mean the probability given by the model might be based off an approximation, it also avoids having an additional bias in the model originating from the specific distribution of the training set. Further thought might however be given to this in order to evaluate the potential benefit of it.

Since the classes we are interested in slightly differ from the ones used in [9], we will have to adapt the hierarchy. The hierarchy used in [9] is reported in the figure 22 in the appendix (taken directly from their paper). I will use the same hierarchy, but with our classes instead of theirs.

They also used a pretraining phase, with only the case where the parent class is present. I will first try without this pretraining phase to see if it is necessary. I will apply it later on the CCSMTL's dataset.

Finally, an important part to mention is that the bayesian probabilities are only calculated during the inference phase. Therefore, this modification does not affect the training of the model.

As it turns out, we still see great results

11.2.3 JFHealthcare - Probabilistic-CAM

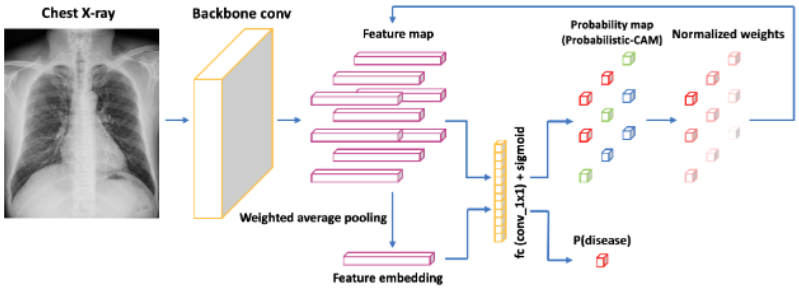


Figure 1: The framework of Probabilistic-CAM (PCAM) pooling.

Figure 18: The figure 1 from [4]

The JFHealthcare team [4] used a probabilistic-CAM method to obtain their results. The method is described in the figure 18. They basically run the feature extraction layers as usual, but add a final convolutionnal layer to obtain a final probability vector, associated with each of the feature maps, as the probability of the class being present in the image. They then use the probability vector to weight the feature maps, and use the weighted feature maps to do a weighted average pooling of the feature maps. The final result is then passed through fully connected layers to obtain the final prediction [22].

They first start by proceeding as usual, applying a feature extraction backbone, followed by a 1x1 convolutionnal layer. They then apply a sigmoid activation function to the output of the fully connected layer, in order to obtain a probability between 0 and 1.

These classes probability are then used as the probability of a disease being present in each of the feature maps. The feature maps are then weighted according to these probabilities using a weighted average pooling. This is then followed by a fully connected layer (with a sigmoid activation), which is used to output the final prediction.

This adds two benefits . The first one is to enable a form of self-attention within the model which they have shown increase the accuracy of the model. But further than that, it also allows for a better understanding of the model's decision. Indeed, the weighted class activation maps offer a more accurate way to visualize the model's decision and potential object location within the image. This is especially useful for the radiologist, who might want to understand why the model made a certain decision.

[22] While it is typical to only have one fully connected layer for the classification, they did not justified their choice of having one layer per class. However, since its a multilabel classification problem, it does not alter the ability of the network to do its job

Experiences	Baseline		Hierarchical		Weighted pooling		CCSMTL		Vingroup Big Data (*)		Radiologists estimated performance	
Metrics / Classes	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
Cardiomegaly	0.74	0.60	0.82	0.64	0.65	0.52	0.82	0.44	**0.86**	-	-	0.678
Pleural Effusion	0.90	0.65	0.91	0.75	**0.93**	**0.75**	0.87	0.61	0.92	-	-	**0.737**
Pneumothorax	0.87	0.22	0.94	0.00	**0.98**	**0.4**	0.84	0.47	-	-	-	-
Lung Opacity	0.88	0.83	0.90	0.84	**0.95**	**0.89**	0.67	0.53	-	-	-	-
Atelectasis	0.67	0.47	0.75	0.63	0.78	0.64	0.74	0.39	**0.83**	-	-	**0.692**
Lung Lesion	0.26	0.67	-	-	-	-	0.71	0.21	-	-	-	-
Pneumonia	0.70	0	**0.88**	**0.10**	0.77	0.11	-	-	-	-	-	-
Consolidation	0.75	0.45	0.88	**0.37**	0.80	0.31	0.71	0.26	**0.94**	-	-	**0.385**
Edema	0.86	0.55	0.70	0.36	**0.94**	0.64	0.77	0.32	**0.93**	-	-	**0.583**
No Finding	0.89	0.93	0.84	0.94	**0.90**	**0.94**	0.82	0.97	-	-	-	-
Mass	-	-	-	-	-	-	0.76	0.27	-	-	-	-
Air Infiltration	-	-	-	-	-	-	0.81	0.49	-	-	-	-
Liquid Infiltration	-	-	-	-	-	-	0.83	0.56	-	-	-	-
Fracture	-	-	-	-	-	-	0.72	0.21	-	-	-	-
Emphysema	-	-	-	-	-	-	0.82	0.38	-	-	-	-
Infiltration	-	-	-	-	-	-	0.70	0.26	-	-	-	-

Table 5: Final results of our experiments. Please note that the results of the Vingroup Big Data experiment are based on their single model results, and not their ensembling results. While not officially first place in the CheXpert results, they did provide the AUC per class - contrary to the other team. The exact configuration of each run can be found in the appendix. Also keep in mind the CCSMTL results refer to another dataset, and are not directly comparable to the other results. The radiologist's estimations come from ChexZero[1]. The exact configuration of each run can be found here 8

11.3 Results on the CCSMTL's dataset

All results can be visualized in Weight and Biases with the proper access.

Following the results obtained earlier on CheXpert, we then tried applying some of the same technique, but this time using a validation dataset from the CCSMTL's data. Sadly, every attempt, no matter what, always seem to converge towards a f1-score of 0.4 , and an approximate AUC of 0.8 (see 19(b) and 19(b)) (the maximum F1-score ever achieved was of 0.42, and the maximum AUC was 0.80) . Since we obtained a way higher score on CheXpert, and even using weights trained on it didn't result in any improvement, we had to face the reality that we most likely had too much noise in our labels. Now , while there are ways to deal with noisy label (some that we will discuss/point later), we do need to have a validation/test dataset free of this noise to evaluate our model's performance on. Sadly, we were not able to obtain help from the radiologists working in the CCSMTL to clean the labels, and while other dataset exists, they do not focus exactly on the same label of interest, and they will not allow us to test the performance of our model on the hospital's data before deploying the model!

There is also the issue of cross-dataset bias. As demonstrated in the section data analysis, there is a stark difference between the data of CheXpert and the one from the CCSMTL. This bias is not only in the form of the label count, but also in the label interpretation . Indeed, there is some variation as to the definition of specific label that varies regionally, based on the radiologists training [28].

I tried starting from the top, verifying every hypothesis, retesting different model, etc. but to no avail. While technique that improved the results on CheXpert did seem to yield better results on the CCSMTL's dataset, they were still not good enough to be considered as a viable solution. An interesting correlation is with the result that were previously obtained by the CheXnet paper. Indeed, they used an automated labeler similar to ours, and proceeded to split the dataset into training and validation without any help from radiologists. Doing so, they too obtained results that were maxing out at 0.435 f1-score and 0.84 AUC [23]. This is a strong indication that the problem is not with our model, but with the dataset itself. The results on the CCSMTL dataset can be seen in the table 5.

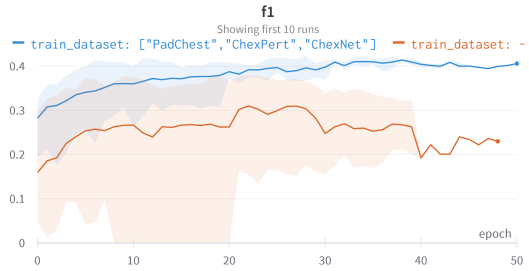
I have briefly attempted to use self supervised method (based on this paper and code [29]) to try and relabel the data for the validation, but without a test set to compare to , I am navigating blindly. I've also tried using this dataset to train a model, and validate on another dataset such as CheXpert. This strangely leads to better results than the one achieved on the CCSMTL's dataset. This would seeem to indicate again that our validation dataset is problematic. While we could easily think that simply using the CheXpert validation set would solve our issue, we would still need a test set to confirm these results before releasing the model, but most importantly, we have been asked to provide different labels than the one provided in CheXpert.Please note I have tried to resample a validation dataset, only to obtain similar results. For now, we could limit ourselves to the labels provided in CheXpert. Doing so, we obtain the results in the table 6. As can be seen, these results are way moore interesting than the one obtained on the CCSMTL's validation dataset, and clearly demonstrate the model's ability to learn through the noise contained within the training set, and it confirms our hypothesis about the issues contained in the validation set [24].

[23] This score was however achieved on a test set that was validated by four radiologist. These radiologist however only worked with low quality images.

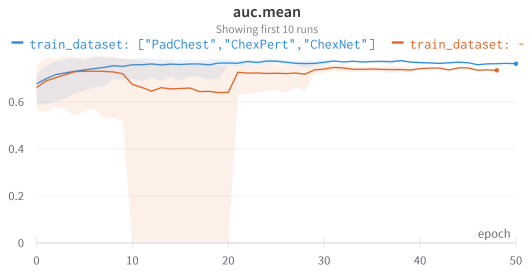
[24] There could also be issues with some of the specific class selected by our radiologists , e.g they could be harder to learn.

Classes	Metrics	
	AUC	F1
Atelectasis	0.76	0.60
Cardiomegaly	0.81	0.64
Consolidation	0.74	0.30
Edema	0.92	0.49
Enlarged Cardiom.	0.39	0.67
Lung Opacity	0.91	0.82
Pleural Effusion	0.93	0.71
Pneumonia	0.71	0.08
Pneumothorax	0.87	0.10
No Finding	0.89	0.94
Mean	0.79	0.54

Table 6: Results on the CheXpert validation dataset obtained from training only on the CCSMTL's dataset. The model was trained for 20 epochs, with convnext large and both the hierarchical and weighted pooling as described earlier. The exact details of this run can be found here https://wandb.ai/ccsmtl2/Chestxray/runs/2zmrtbpv/overview?workspace=user-ccsmtl



((a)) average f1-score on the validation dataset during training



((b)) average AUC on the validation dataset during training

Figure 19: Comparisons of the results on the CCSMTL's dataset with and without additional data for training. The runs were grouped and averaged, with the min-max values being displayed as the range. Train dataset = : - reflects that no other dataset than the CCSMTL's was used for training, as it was always used by default.

12 Explainability & Visualization

I used GradCAM [30] to visualize the model's decision. The GradCAM method is a method that allows to visualize the regions of the image that the model used to make its decision. It does so by computing the gradient of the output of the model with respect to the last convolutional layer, and then using this gradient to weight the feature maps. The weighted feature maps are then summed up to obtain a single feature map, which is then used to obtain the final heatmap. The final heatmap is then used to overlay the heatmap on the original image, in order to visualize the regions of the image that the model used to make its decision.

I compared a few of the different method offered by the GradCAM package, as listed in the

table 7. Sadly , neither the CCSMTL's data nor the CheXpert dataset provides image-level annotations. I therefore had to rely on qualitative analysis for now. As can be seen in 20(a), while some method are simply to discard (pointing you "EigenCAM"), others do provided interesting results. While on first sight the ScoreCAM method seems to be more precise than the GradCAM (vanilla) or HiResCAM, it does not always correlate with them.

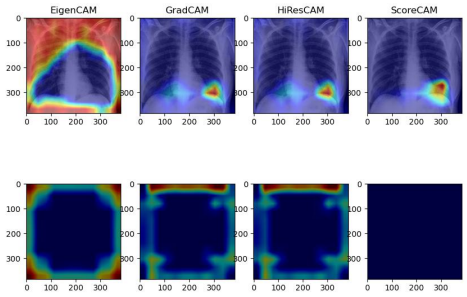
Also interesting to notice, the model seem to have identify an object in 20(b). While the label identify this patient as No finding, the model predicts "insert prediction", which could explain the visual we obtain. We would need the expert opinion of a radiologist to determine whether our model is right or wrong however, but since thee CheXpert validation dataset was reviweded by three radiologist, it is more likely that this is a false positive.

To determine which is better, we will need to quantify these results. Sadly, I was out of time to achieve this goal. Ideally, we would want to use the CCSMTL's dataset to do so, but as mentionned earlier, the CCSMTL has difficulty providing radiologists time for the project. My supervisor Jean-Francois Samson therefore found another dataset, VinBigData[6], that does provide image-level annotations. We hope to soon use this dataset in order to evaluate the mean Average Precision (mAP) [31].

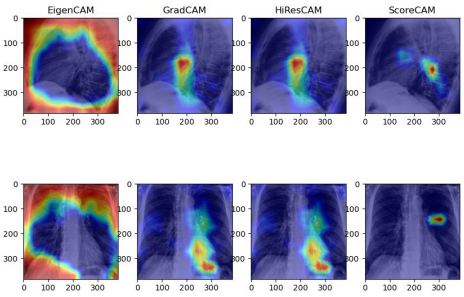
Other examples of the results can be found in the appendix at 23(a) and 25(a).

Method	What it does
GradCAM	Weight the 2D activations by the average gradient
HiResCAM	Like GradCAM but element-wise multiply the activations with the gradients; provably guaranteed faithfulness for certain models
GradCAMElementWise	Like GradCAM but element-wise multiply the activations with the gradients then apply a ReLU operation before summing
GradCAM++	Like GradCAM but uses second order gradients
XGradCAM	Like GradCAM but scale the gradients by the normalized activations
AblationCAM	Zero out activations and measure how the output drops (this repository includes a fast batched implementation)
ScoreCAM	Perbutate the image by the scaled activations and measure how the output drops
EigenCAM	Takes the first principle component of the 2D Activations (no class discrimination, but seems to give great results)
EigenGradCAM	Like EigenCAM but with class discrimination: First principle component of Activations*Grad. Looks like GradCAM, but cleaner
LayerCAM	Spatially weight the activations by positive gradients. Works better especially in lower layers
FullGrad	Computes the gradients of the biases from all over the network, and then sums them
Deep Feature Factorizations	Non Negative Matrix Factorization on the 2D activations

Table 7: Different methods offered by the GradCAM package. This table is taken verbatim from the github repository for GradCAM [25].



((a)) Example 1



((b)) Example 2

Figure 20: Two examples of patient files being interpreted by the model using GradCAM.The first one is a patient with an Edema, and the second one is one that is supposed to be healthy.

I, however, am quite satisfied that the model's deccision is not only based on the lung's part of the image, but that it also seem to coincide with the region of interest (ROI) that the pathologies would normally be. While I am no expert in radiology, based on my understanding, the model does seem to draw a plausible ROI.

Finally, however, this method is not subject to any regularization, since the model has only been trained on classification, and that these region are inferred from the model's decision. A future step in this project would therefore be that, once our best model has been trained (ideally by doing an ensemble of model), to use the best GradCAM method (as determined by the mAP) in order to generate pseudo labels. These pseudo-labels would then be used to train an object-detection model such as yolo[20], which would hopefully learn to detect the pathologies in the image better than our original model ensemble, but also, importantly, faster. Currently, some method such as ScoreCAM does require a lot of computationnal ressources (For one image, it require about 1 minute on a 3090 GPU for a single model), which would lead to a bottleneck for the deployment of the model.

You will also noticed that we tested the empty images on the GradCAM. While the model is training/infering, these empty images, used as padding to facilitate the job of the dataloader, are not given to the model. However, it is still interesting to see how our model reacts when it sees them, demonstrating any "bias" towards a specific region of the image.

13 Failed Attempts

13.1 Moving Average Labels

While we know we have many errors in our dataset, it is our hope that our model will simply ignore this as noise. However, there are more advanced techniques to deal with noisy labels in your training data. Based on the " Deep learning with noisy labels: exploring techniques and remedies in medical image analysis [32]" and " Learning from Noisy Labels with Deep Neural Networks: A Survey [33] ", I've tried implementing step mechanism that would modified the training label, applying them as a moving average such that, each time a prediction p_i for a given label L_i, is made by the model, the label is modified to

$$L_{i+1} = \alpha L_i + (1 - \alpha) P_i \tag{5}$$

I've initially tried setting the α parameter to 0.999 but, immediately, runs done with such mechanism led to overfitting. Any further attempts hasn't yielded any interesting results either. This solution was therefore discarded

13.2 Polycam (PCAM)

While the Gradient Class Activation (GradCam) methods allow to visualize the most important regions of the image for a given class, it is quite imprecise. A lot has been done towards providing a better explainability for deep learning models, and one such paper is the Polycam paper [34]. The paper proposes a method to obtain a more precise localization of the class-related pixels, by leveraging the information present at multiple layers throughout the network. Doing so , they did improved by a lot the precision over the GradCAM method. However, our attempt to use it has failed. For now, we could only obtain noise from the method, and we were unable to find a way to improve it. We will try to come back to it later.

13.3 Adding Data

I tried supplementing our dataset with data from different sources, mainly MimicCXR [11] .However, this didn't led to the increase in score that I hoped. I think it is because of the difference in interpretation between the two datasets. As described in [28], radiologists from different region in the world can interpret the same image differently. This means combining two datasets together will simply lead to more noise in the data. Without having a radiologist certifying that the label from the two sources are equivalent, combining two or more datasets together is probably not the best idea.

13.4 Optimizers

I also tried other optimizer, to see if they would generalize better. Among other things, i've tried RMSprop, Adagrad, Adam and SGD (with momentum). However, none of them seemed to perform better than AdamW

13.5 Learning rate schedulers

Just like with the optimizer, I wanted to validate my choice of scheduler. Doing so, I've tested the following schedulers: CosineAnnealingWarmRestarts, CosineAnnealingLR, OneCycleLR and StepLR. Again, OneCycle learning rate did seem to provide the best results.

14 Problem observed

A multitude of problems slowed down the development of the model. I will try to cover the most important ones here, and the lesson learned from them.

14.1 Automatic Mixed Precision (AMP)

The training on such a massive dataset is rather slow. To solve this problem, I tried to implement the automatic mixed precision (AMP) technique available with pytorch. This technique is a way to

train a model in a lower precision, while still using a higher precision for the gradient computation. This is especially useful when training on a GPU, as it allows to reduce the memory usage, and therefore to train on larger batch size, by applying a mix of 32 bits precision and 16 bits precision multiplication.

However, this technique is not without its problems. While it improved the training time vastly, it did lead to some unstability, with the gradient sometimes underflowing and becoming NaN.

This is a problem that I have mostly solved by normalizing further the input vector but some underflowing might still occur. In these case, the training need to be done again as the model will be corrupted (Nan values in the weights).

14.2 Normalizing the data

While the data was at first simply normalize by the mean and standard deviation of the ImageNet dataset. However, we realized this was not enough . This easily led to unstability in the training process, and we had to further normalize the input. The first attempt at this was with a min max normalization to rescale between 0 and 1, before applying the ImageNet's normalization .While this solved the issue, we did try to further improve the normalization by using the CLAHE algorithm. This algorithm is used to improve the contrast of the image, and is often used in medical imaging.

14.3 Verifying the data

The data was not as cleaned as we would have liked. While the CheXpert validation dataset was validated by a team of three radiologist, our data was simply annotated by a single radiologist's report, from which the labels were then automatically derived. This lead to a quite important source of error as radiologist's performance is estimated to be between 0.4 and 0.6 [1]. This is a quite important source of error, which is compounded by the fact that the radiologist's report often contained many uncertainties that we had to quantify with an automated labeler. The problem's come that no radiologist was available to verify the work of the labeler, and we had to rely on our interpretation of the report to evaluate the performance of the rule-based labeler we developped.

While it usually would simply be solved by manually annotating a few examples and compare the results accrosss the manually labeled dataset and the automatically labeled one, this was not possible in our case. Indeed, the nature of our medical dataset required an expert eye to verify the data, and even then, radiologists often disagree on the label of a given image. This is why when working on medical data, it is important to have a team of specialist available to support the project.

14.4 Drivers & CUDA

Many times did we had issues with our drivers. This happened because we were updating the packages as we were developping the project, and because we were working on parallel on more than one project on the same machine. To rectify and avoid this, a simple solution is to use a virtual environment, which will allow us to have a clean environment for each project, and to avoid any conflict. Docker is a good solution to provide such virtual environment. However, we had issues installing and using Docker properly, and we managed to resolved our initial problem in the meantime.

14.5 Multi-GPU training

To train on multiple GPU, I first used the DataParallel module from PyTorch. However, this module is deprecated, and I therefore chose to switch to the DistributedDataParallel module. This module is more efficient, but it is also more complicated to use. Just as to prove this point, it often break down, seemingly at random, and I had to restart the training process.

While in and of itself it would only be annoying, the real problem is that the training would not simply stop, but would not converge, leading to wrong conclusion when trying to compare the results of the different models. It seems like the problem might arise from the synchronization of the gradient between the batches on each GPU. You see, compared to data parallel, the distributed data parallel instead of splitting the model between the GPUs, it splits the data between the GPUs. This means that the gradient are computed on each GPU, and then synchronized between the GPUs.

This is a problem that I have not been able to solve, and I have not been able to find any solution to it online. I have therefore decided to switch back to the DataParallel module, which is less efficient, but at least it works.

This problem was left unnoticed for a while and resulted in a lot of wasted time. This is why it is important to always check the results of your model, and to always compare them to the results of the previous model. If I had compare the single run with the multi-gpu run from the start, I would have notice that the training can converge for the single run, and not for the multi-gpu run. However, this problem seemed to be on and off and was very hard to detect at first.

14.6 Data augmentation

At first, I tried to implement a few technique of data augmentation of my own, as the torchvision library does not provide a lot of data augmentation technique. However, I quickly realized that the data augmentation technique I implemented were not as good as I wanted, being both not optimized enough and containing errors I would catch from time to time.

Later, I found the Albumentations library [15], which is a library that provides a lot of data augmentation technique, and is very easy to use. I therefore decided to switch to this library, and it has proven to be a very good choice. It allowed me to quickly and easily test a few different data augmentations without worrying about potential errors in my implementation.

However, it is not idiot-proof either. As I was working on the project, I later found a bug where my image input of format (C,H,W) was not being converted to (H,W,C) by the library, as it required, but was still processed, leading to disastrous images.

To avoid this step, a jupyter notebook was then used to verify from now on the output of our dataloader at different step in order to verify the integrity of the image throughout its pre-processing and data-augmenting journey.

14.7 Unitary Tests

Based on the experience described above, I also decided to implement unitary testing in order to limit the errors that could be introduced in the code. The master branch of the git repository was also linked to these unitary tets such that any pull requests would first have to pass the tests before being merged into the branch. This is especially useful when working with a group, but even when working alone it is a good way to ensure that the code is working as intended and catch unintentional errors.

14.8 Unstability during training

14.8.1 Gradient's clipping

While we already experienced some unstability before, we mostly tried to solve it by normalizing the input. However, we still had some issues with the training process. To avoid overflowing gradients, we tried to use the gradient clipping technique [35]. This technique is a way to limit the gradient to a certain value, in order to avoid them from exploding.

By default, we set the maximum gradient norm to 1, but we also tried to set it to 2, and to 5. However, we found that the best results were obtained with a maximum gradient norm of 1. This also serves as a way of regularizing the model, as it forces the gradient to be small, and therefore forces the model to learn slowly.

15 Deploying the model

15.1 Backend

The backend of the application is written in Python, using the Flask framework. The backend is responsible for the communication between the frontend and the model. It is also responsible for the preprocessing of the images, and for the prediction of the model.

The backend is deployed on a server, and is accessible through an API.

15.2 Frontend

The frontend of the application is written with the Dash framework [36]. This framework is a wrapper around the Flask framework, and allows to create a web application.

This web application is accessible through a web browser, and allows the user to upload an image, and to get the prediction of the model. It also allow to use a few example preloaded in the server. This will allow us to easily display the capacity of the model, and to gather feedback from the user.

Please give the appropriate credit to the authors of the template use to developped the web application. [26]

Once its effectiveness had been demonstrated, it was decided to give the task of developping a cleaner, more professionnal and efficient interface to a team of web developper. This allowed the team to shift back its focus on the machine learning aspect of the project. An example of the interface is shown in Figure 26.

The interface allows to quickly see which pathology has been predicted by the model, thanks to the color-coded graph shown in the bottom-right corner of the interface. The disease predicted by the model have a button appeared in the right section, which, when the cursor is placed on top, will display the corresponding heatmap above the image. This allow for the user to quickly see the image, unhindered by the heatmap, and to quickly inspect all the different heatmaps provided by the model. This speed of use is an important requirement, since the radiologists spends very little time per image, and go as far as to count each and every click of mouse to avoid wasting time.

Finally, the interface also show the labeled of the image, by displaying the name of the class in red in the aforementioned graph. This allow us to quickly compared the model's prediction to the ground truth.

[26] It was developped by the Dash team and is available at https://github.com/plotly/dash-sample-apps/tree/main/apps/dash-image-processing

16 Exploration

This section covers later attempts that, while seeming successful, I had not the time to dwelve in deep enough to be able to include them in the final report. I will however quickly mention them here for future reference.

16.1 Ensembling

Since we had already trained a few successful and different models, I attempted a simple ensemble of the models. I empirically found that the best results were obtained by not averaging the results, but taking the minimum of the results. This seems to indicate our model might have a preponderance towards false positive.

However, the results were only slightly better by taking an ensemble. I achieved an average AUC of 0.84 and an average F1-score of 0.60 doing so.

16.2 Pretraining

Since we also had a lot of other dataset (Mimic-CXR [11],VinBigData [6], NIH Chest X-Ray [2],ours,PadChest [37] and CheXpert), I attempted to pretrain the model on these dataset. Trying to add simply add them during training did not seem to have a significant or noticeable impact on the resutls, probably because of the aforementionned problem with cross-generalization. Howeever, simply pretraining on them could allow us to improve our results. Doing so, I did manage to slightly increase the best F1-score previously obtained.

I achieved an average F1-score of 0.60, with an average AUC of 0.80

16.3 Combining Hierarchical and Weighted pooling

I also quickly tried to combine both the weighted pooling and hierarchical learning methods together as they focused on different aspects. I had hoped that together, they could achieved a better results than individually but I did not have time to explore this option thoroughly. For now, it did not seem to yield better results.

17 Future Steps

I have identified the remaining step of the project before deployment as being :

1. Create a validation and test set for the CCSMTL's dataset with the help of radiologists
2. Clean the training data based on the feedback from the radiologists
3. Find the best GradCAM method based on the mAP score on either the CCSMTL's improved dataset, or on vinBigData
4. Create a model ensemble from the different model/training method explored that performed successfully
5. Train Yolo (or another detection model) from the pseudo labels generated by the model ensemble

6. Filter the data sent to the model, either by training a Unet or by other method, to avoid ,broken, "weird" images to make it to the model to preserve the trust of the radiologist using it.

7. Use the feedback from the radiologists to improve the model , change the categories, expand or reduce the number of categories, etc.

8. Create a final ensemble of model

9. The most important, final step : test the model on a validated test dataset from the CC-SMTL's hospitals before pre-releasing the model

10. Deploy the model as fully operationnal

11. Write a protocol to verify the model's performance on a regular basis, in order to prevent model's drift. (This could be the result of changes in the radiologist's practice, the introduction of new X-ray machine, etc)

18 Critical Evaluation of the internship

While this internship was a great learning opportunity, there were quite a few mishaps that could have been avoided. I will do my best to organize my thoughts and explain the issues I encountered, and how I could have avoided them.

18.1 Delays

The first issue that arise with this internship was the delay in the start of the internship. When I first arrived in May, they had unanticipated delays in the reception of their server, meaning we had no computationnal ressources. It took a bit over a month before the server was finally delivered, and we were able to start working on the project.

After that, more delays in the reception of their data was not communicated with me before my internship, and I was informed after I had started that the data that I was supposed to be working on would arrive partially throughout summer, only to be received in full by the end of August. This meant that I had to work on the CheXpert dataset for a month longer than expected, and that I would only have two months of my internship to work with the CCSMTL's data.

This was a disapointment, and showed a lack of communication from the CCSMTL's side, as I was never warned about the delays and the previous information I was given was that everything would be ready for the start of my internship.

But, just as with everything in life, most thing do not go according to plan. I've therefore tried to make due with what I had, taking this month to prepare the code, explore data augmentations technique and do a thorough litterature review, in the hope this preparation would compensate for the reduced time I had to work on the project.

18.2 Technical Issues

The biggest technical issues I've had throughout the project arose all from the same source : my lack of verification. Since I was working on my part of the project alone, I had to verify everything by myself.

Sadly, I did not initially do a very good job at it , and I would later discover slight errors in my code that would invalidate my results. This would lead to me having to redo the whole process, which was very time consuming. In order to avoid those errors in the future, I've learn to develop unit tests for my code, and to do code reviews with my colleagues. This would allow me to catch those errors before they make it to the training phase. I've implemented automated test using the pytest library, which would be launch everytime a pull request was done on the main branch ; blocking the merge if the tests failed. This would ensure that the code was always working, and that I would not have to redo the whole process again.

I've also started to use jupyter notebooks to verify part of my code, as unit test can only catch errors the break the code ; not conceptual errors that would lead to wrong results. This verification step was crucial to assert the validity of the data augmentations that I was using for example.

18.3 Project Management Issues

While I was working at the CIUSSS Centre Sud, I was mostly left on my own to work on the project. This was a great learning opportunity, and at first I did appreciate it, as I've learned to work on my own, and to manage my time. However, I've quickly realized that I was lacking experience, which slowed down tremendously my work.

Sadly, my supervisor at the CCSMTL was rarely available to help me. This led me to develop a bad habit of not documenting my work, and not organizing my thoughts. To correct for this, I would work more closely with the other intern, asking them to review my pull requests, in the hope of avoiding any mistake and making sure my code was understandable. I would also try to document my work more, and to organize my thoughts before starting to code. I made good use of the project management tool of Azure Devops, in order to create Epics,features, User Stories and etc. to organize my work and tasks. This would allow me to keep track of my progress, and to make sure I was not forgetting any important step.

This lack of supervision also meant it was left to the three of us working on the project to find ourselves tasks to work on. This forced me to develop a great autonomy, having to elaborate what would be the next steps on the project by myself.

18.4 Interpersonnal Issues

Finally, the last issue I've had with this internship was with some interpersonnal conflict with one of the employee at the CCSMTL. While working there, one employee would often make me feel uncomfortable, playing with my hair , fondle me , massaging my shoulder, etc. While I mentionned my incomfort vocally multiple time, this person would not stop. I've tried to ignore it, but it was becoming more and more difficult to work with this person. I was also not the only person made uncomfortable by this person, as I've heard the other intern complain multiple time to her about her comportment.

While I was supposed to work on the project with this person, she also showed a lack of experience and knowledge about machine learning, data science and programmation that made it hard to collaborate with her. I would spend most of my day explaining basic concept of programmation to her, and her lack of independance and autonomy meant she would often ask me to do her work for her, or to closely supervise her all day long. While I tried at first, it was clear that I would not be able to complete my task if I were to teach her everything she needed to know. While I tried to talked with my supervisor about the issue, I failed to notice any action to resolve the problem. The only mention he made of this was to tell me that I should teach her more about the subject, to make it easier for her to work with us, but didactic material that I've sent her (tutorial, online videos, online classes, etc.) would not be used by her.

In the end,her lack of respect for my physical boundaries, combine with her lack of autonomy meant I had to simply remove myself from any contact with this person, in order to be able to continue to work properly, and leave her to work on her own.

While this was an unpleasant experience, I was left even more disapointed by the reaction from the CCSMTL and my supervisor, who simply complained about my inability to work in a team, ignoring the source of the problem. I felt particularly sadden by my supervisor, who simply called me immature for my reaction.

However, I still try to focus on the bright side, and I've learned a lot from my internship, and I was able to work great in collaboration with the other intern. I've learned that I should not be afraid to speak up if I feel uncomfortable, even if it might be difficult to do so or if you might not be supported in your decision to do so.

While not everything worked out the way I had hoped, this internship allowed me to work on computer vision on a scale student's rarely have the chance to do. It also allowed me to touch at every step of a ML project, from the data acquisition to the deployment of the model.

The only truly regrettable part was the ïncident ẅith the employee, and the unability to work with a specialist to validate our data, which I have no doubt would have made a great difference in our results.

19 Conclusion

while everything was not perfect, this internship offered me an opportunity to get my first work experience in a machine learning project. I've learned a lot about the field, and I've also learned a lot about myself, and how to work in a team. While we did not succeed to develop a ready-for-production model, we did managed to reproduce the CheXpert results, obtaining an F1-score of 0.59 on the validation set, and an AUC of 0.82. Even if we did not succeed to reproduce the same results on the CCSMTL's data and classes (which I believe would have been possible with a better validation process),obtaining an AUC of 0.80 and F1-score of 0.42. I am still proud of the work we've done, and I am confident that it can be reused in the future to develop a better model, once the CCSMTL manages to obtain time from their radiologists to review a test/validation dataset.

Not only did I managed to reproduce the CheXpert's competition results, I also managed to develop an interface to present the model, helped another intern curate a dataset of 400 000 images, and I tested different techniques to move from the classification task to the weakly supervised object detection one. While I did not had time to complete this part, I outlined the steps that would be needed to complete this task. Thanks to this internship helped me developped my skills in computer vision and programmation, and to increase my confidence in my abilities.

20 Appendix

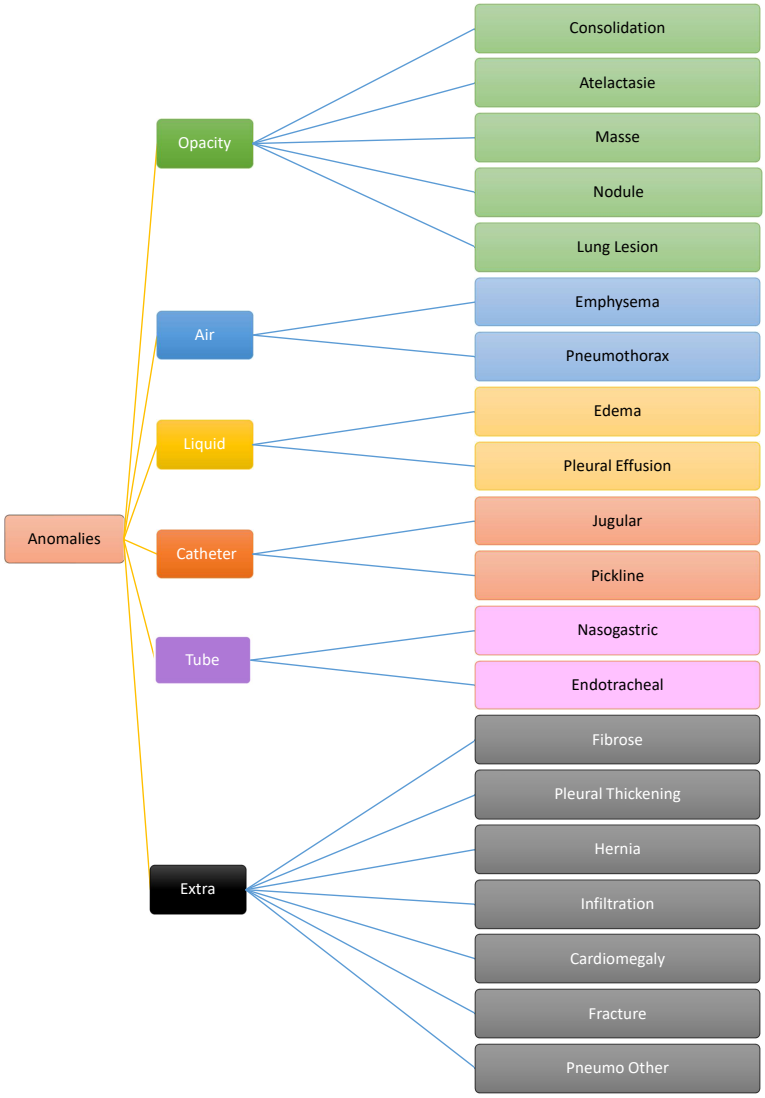


Figure 21: Graph of the different pathologies

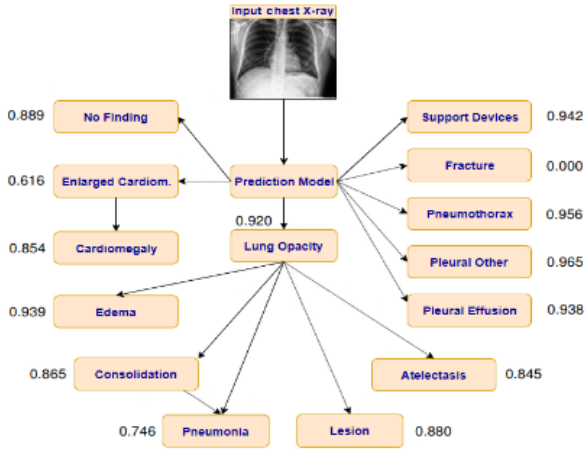
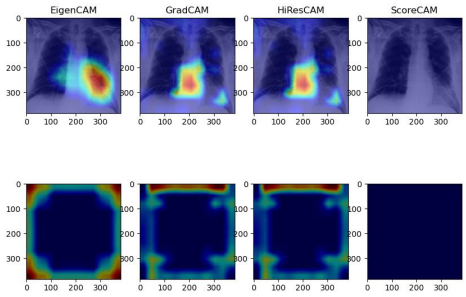


Figure 1: Illustration of our classification task, which aims to build a deep learning system for predicting probability of presence of 14 different pathologies or observations from the CXRs. The relationships among labels were proposed by Irvin et *al.* [21].

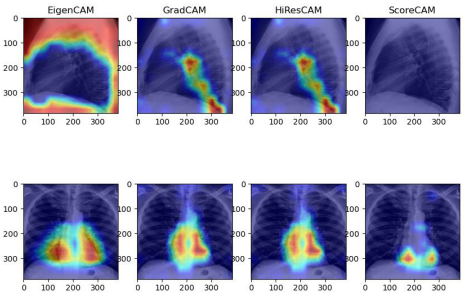
Figure 22: Figure 1 from the paper [9] describing the hierarchy between the label of CheXpert. This figure was produced by another intern, Maxime Fournier

Parameters	Run Configuration		
	Baseline	Hierarchical	Weighted
Weight&Biases	Devoted-plant	Absurd-disco	Stellar-lake
Backbone	ConvNeXt-Large	ConvNeXt-Large	ConvNeXt-Large
Batch size	64	32	32
Learning rate	1e-4	1e-3	1e-4
Weight decay	1e-2	1e-2	1e-2
dropout	0	0.1	0
samples $epoch^{-1}$	50_000	50_000	50_000
Optimizer	AdamW	AdamW	AdamW
Scheduler	OneCycle	OneCycle	OneCycle
Patience	20	40	40
Label smoothing	0	0	0
Clip norm	1	1	1
Epoch (max)	200	200	200
# of Channel	3	3	3
Augmentations prob	[1,1,0.5,1,1]	[0.8,0.8,0.5,0.1,0.1]	[0.8,0.8,0.5,0.1,0.1]
Training datasets	CheXpert	CheXpert+MimicCXR	CheXpert+MimicCXR
Pretraining(# epoch)	0	0	0

Table 8: run configuration for the final results in the table 5

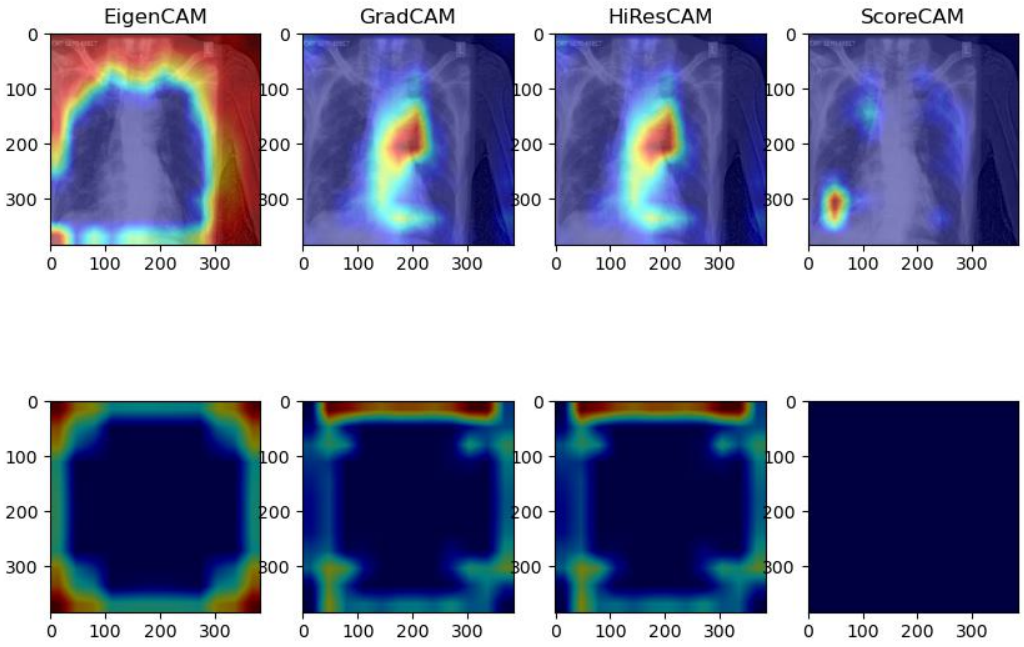


((a)) Example 1

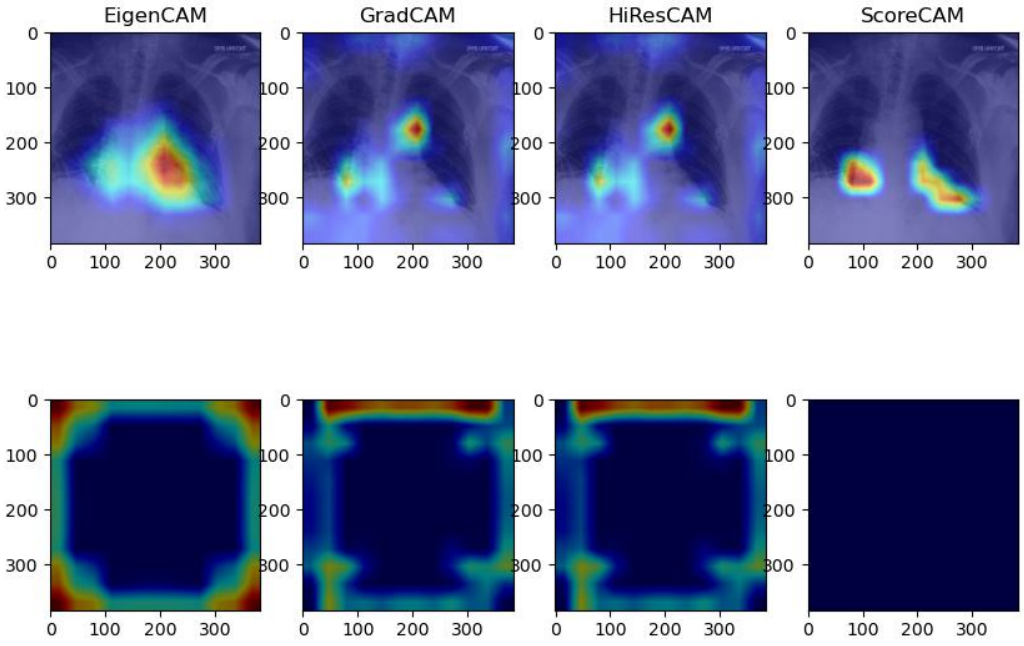


((b)) Example 2

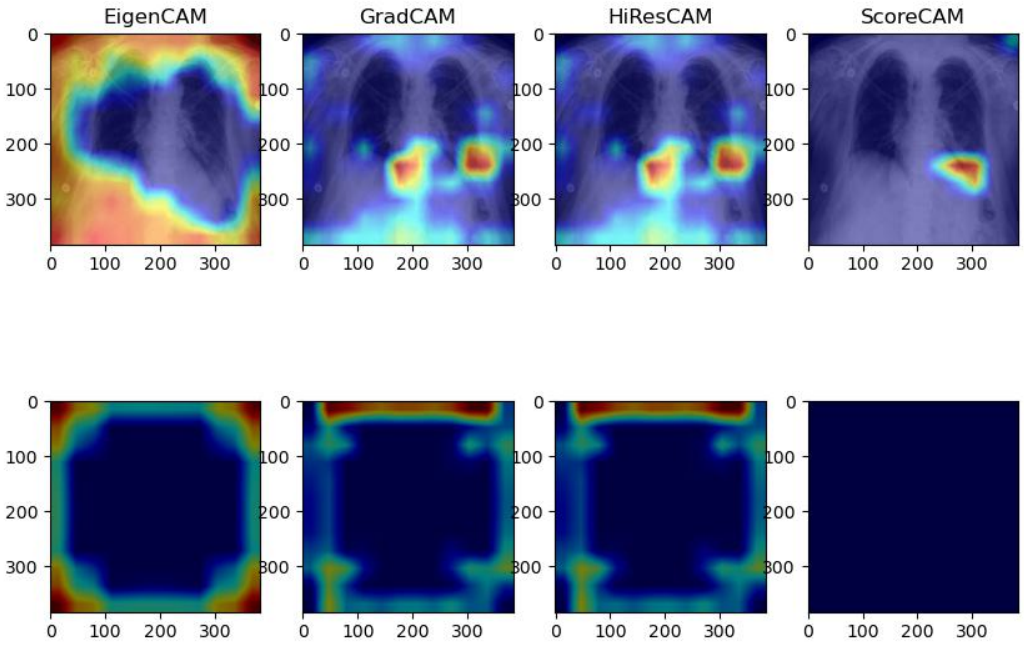
Figure 23: Two examples of patient files being interpreted by the model using GradCAM. The first one is a patient labeled no finding, and the second one is labeled with possible atelectasis, consolidation, cardiomegaly and edema.



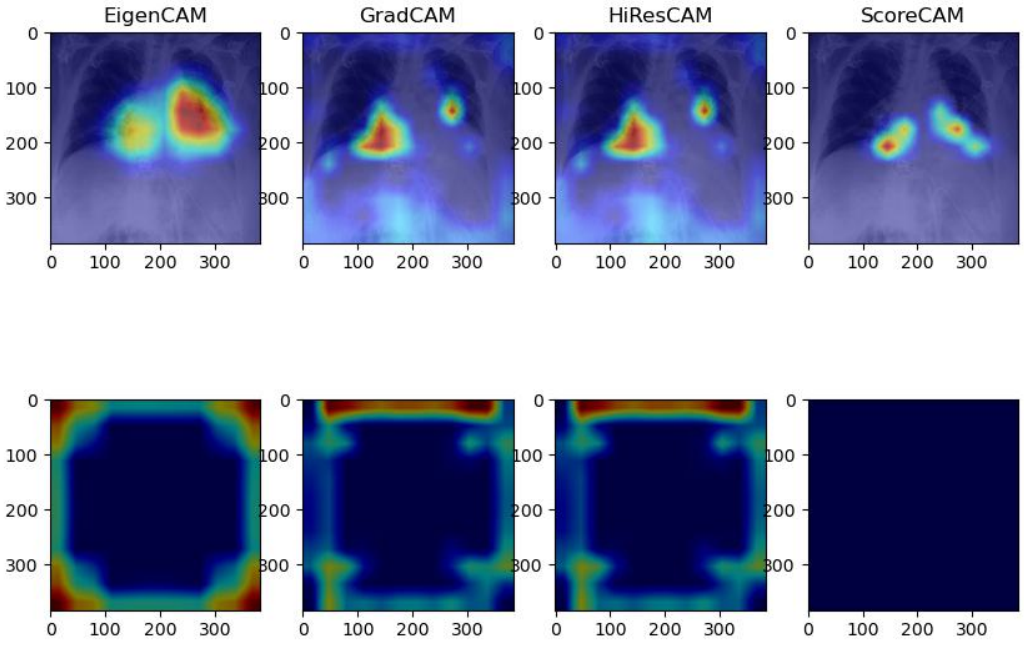
((a)) Example 1



((b)) Example 2



((a)) Example 1



((b)) Example 2

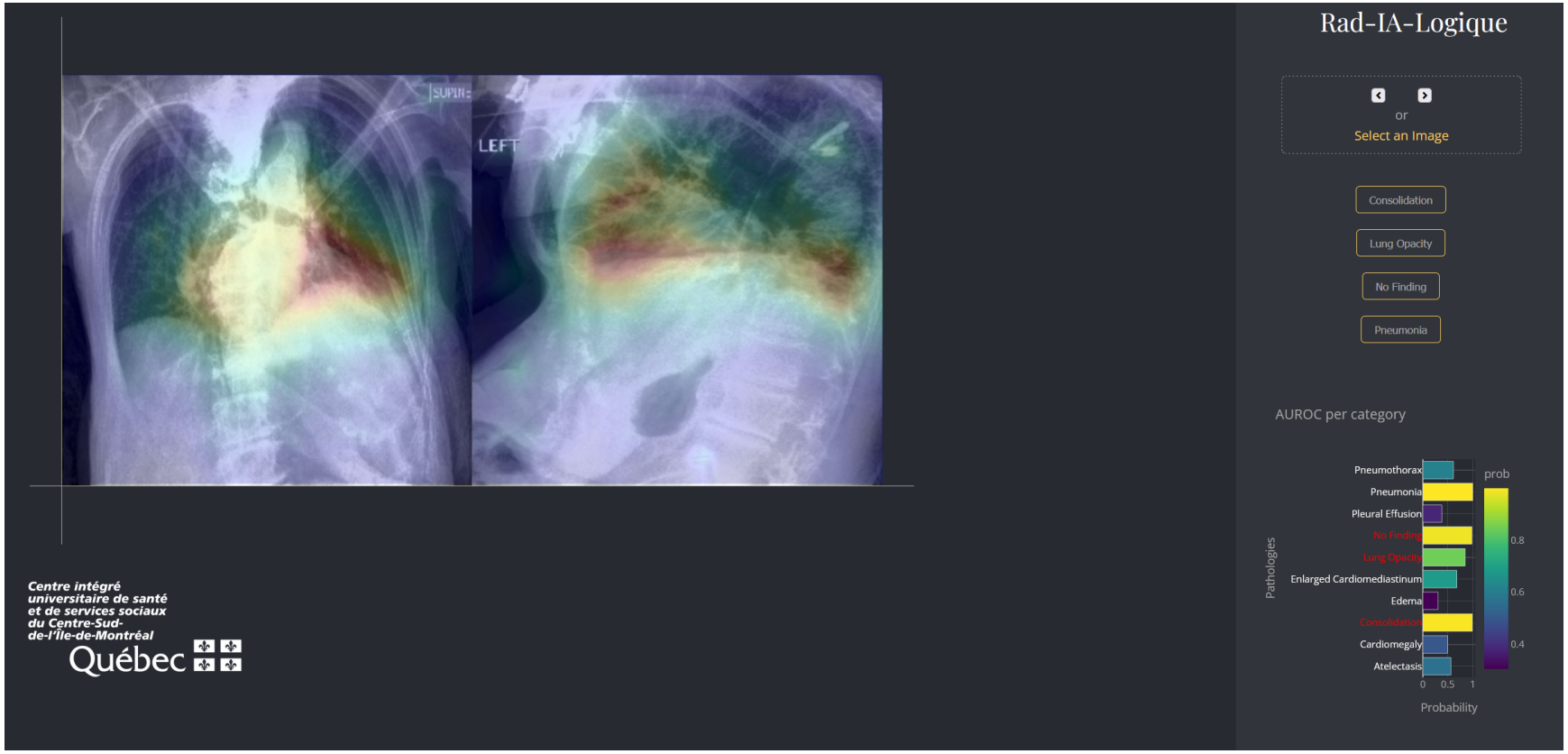


Figure 26: Example of the interface

1. Tiu, E. **andothers**. Expert-level detection of pathologies from unannotated chest X-ray images via self-supervised learning. *Nature Biomedical Engineering* **6,** 1–8 (**september** 2022).
2. Rajpurkar, P. **andothers**. *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning* 2017. https://arxiv.org/abs/1711.05225.
3. Irvin, J. **andothers**. *CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison* 2019. https://arxiv.org/abs/1901.07031.
4. Ye, W., Yao, J., Xue, H. & Li, Y. *Weakly Supervised Lesion Localization With Probabilistic-CAM Pooling* 2020. arXiv: 2005.14480 [cs.CV].
5. Press, D. G. T. C. *Medical imaging backlogs plaguing Canadian hospitals, radiologists warn* https://globalnews.ca/news/8505265/medical-imaging-backlogs-canadian-hospitals/.
6. Nguyen, H. Q. **andothers**. *VinDr-CXR: An open dataset of chest X-rays with radiologist's annotations* 2020. https://arxiv.org/abs/2012.15029.
7. Oakden-Rayner, L. *Half a million x-rays! First impressions of the Stanford and MIT chest x-ray datasets* 2019. https://laurenoakdenrayner.com/2019/02/25/half-a-million-x-rays-first-impressions-of-the-stanford-and-mit-chest-x-ray-datasets/.
8. Oakden-Rayner, L. *CheXNet: an in-depth review* 2018. https://laurenoakdenrayner.com/2018/01/24/chexnet-an-in-depth-review/.
9. Pham, H. H., Le, T. T., Tran, D. Q., Ngo, D. T. & Nguyen, H. Q. *Interpreting chest X-rays via CNNs that exploit hierarchical disease dependencies and uncertainty labels* 2019. https://arxiv.org/abs/1911.06475.
10. Yuan, Z., Yan, Y., Sonka, M. & Yang, T. Large-scale Robust Deep AUC Maximization: A New Surrogate Loss and Empirical Studies on Medical Image Classification. https://arxiv.org/abs/2012.03173 (2020).
11. Johnson, A. E. W. **andothers**. *MIMIC-CXR-JPG, a large publicly available database of labeled chest radiographs* 2019. https://arxiv.org/abs/1901.07042.
12. Biewald, L. *Experiment Tracking with Weights and Biases* Software available from wandb.com. 2020. https://www.wandb.com/.
13. Sitarz, M. *Extending F1 metric, probabilistic approach* 2022. https://arxiv.org/abs/2210.11997.
14. Gösgens, M., Zhiyanov, A., Tikhonov, A. & Prokhorenkova, L. *Good Classification Measures and How to Find Them* 2022. https://arxiv.org/abs/2201.09044.
15. Buslaev, A. **andothers**. Albumentations: Fast and Flexible Image Augmentations. *Information* **11.** ISSN: 2078-2489. https://www.mdpi.com/2078-2489/11/2/125 (2020).
16. Siddhartha, M. & Santra, A. *COVIDLite: A depth-wise separable deep neural network with white balance and CLAHE for detection of COVID-19* 2020. https://arxiv.org/abs/2006.13873.
17. Huang, G., Liu, Z., van der Maaten, L. & Weinberger, K. Q. *Densely Connected Convolutional Networks* 2016. https://arxiv.org/abs/1608.06993.
18. Liu, Z. **andothers**. *A ConvNet for the 2020s* 2022. https://arxiv.org/abs/2201.03545.
19. Tan, M. & Le, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. https://arxiv.org/abs/1905.11946 (2019).

20. Jocher, G. *YOLOv5 by Ultralytics* **version** 7.0. **may** 2020. https://github.com/ultralytics/yolov5.

21. Dosovitskiy, A. **andothers**. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* 2020. https://arxiv.org/abs/2010.11929.

22. Touvron, H. **andothers**. *Training data-efficient image transformers amp; distillation through attention* 2020. https://arxiv.org/abs/2012.12877.

23. Haque, M. I. U., Dubey, A. K. & Hinkle, J. D. The Effect of Image Resolution on Automated Classification of Chest X-rays. *medRxiv*. eprint: https://www.medrxiv.org/content/early/2021/08/01/2021.07.30.21261225.full.pdf. https://www.medrxiv.org/content/early/2021/08/01/2021.07.30.21261225 (2021).

24. Smith, L. N. & Topin, N. *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates* 2017. https://arxiv.org/abs/1708.07120.

25. Loshchilov, I. & Hutter, F. *Decoupled Weight Decay Regularization* 2017. https://arxiv.org/abs/1711.05101.

26. Kuhn, L. *Faster Deep Learning Training with PyTorch – a 2021 Guide* https://efficientdl.com/faster-deep-learning-in-pytorch-a-guide/.

27. Wightman, R. *PyTorch Image Models* https://github.com/rwightman/pytorch-image-models. 2019.

28. Cohen, J. P., Hashir, M., Brooks, R. & Bertrand, H. *On the limits of cross-domain generalization in automated X-ray prediction* 2020. https://arxiv.org/abs/2002.02497.

29. Cohen, J. P. **andothers**. *TorchXRayVision: A library of chest X-ray datasets and models* **in** *Medical Imaging with Deep Learning* (2022). https://github.com/mlmed/torchxrayvision.

30. Selvaraju, R. R. **andothers**. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision* **128,** 336–359. https://doi.org/10.1007%2Fs11263-019-01228-7 (**october** 2019).

31. Otani, M. **andothers**. *Optimal Correction Cost for Object Detection Evaluation* 2022. https://arxiv.org/abs/2203.14438.

32. Karimi, D., Dou, H., Warfield, S. K. & Gholipour, A. Deep learning with noisy labels: exploring techniques and remedies in medical image analysis. *CoRR* **abs/1912.02911.** arXiv: 1912.02911. http://arxiv.org/abs/1912.02911 (2019).

33. Song, H., Kim, M., Park, D. & Lee, J. Learning from Noisy Labels with Deep Neural Networks: A Survey. *CoRR* **abs/2007.08199.** arXiv: 2007.08199. https://arxiv.org/abs/2007.08199 (2020).

34. Englebert, A., Cornu, O. & De Vleeschouwer, C. *Poly-CAM: High resolution class activation map for convolutional neural networks* 2022. https://arxiv.org/abs/2204.13359.

35. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, 2016).

36. Hossain, S. *Visualization of Bioinformatics Data with Dash Bio* **in** *Proceedings of the 18th Python in Science Conference* (eds Calloway, C., Lippa, D., Niederhut, D. & Shupe, D.) (2019), 126–133.

37. Bustos, A., Pertusa, A., Salinas, J.-M. & de la Iglesia-Vayá, M. PadChest: A large chest x-ray image dataset with multi-label annotated reports. *Medical Image Analysis* **66,** 101797. https://doi.org/10.1016%2Fj.media.2020.101797 (**december** 2020).