**Machine Learning Engineer Nanodegree**

**Capstone Project: Pneumonia detection based on chest X-Ray Images**

François Masson  
October 2018

**I. Definition**

**Project Overview**

Pneumonia is the eighth leading cause of death, and the number 1 cause of death from infectious disease, in the United States. About 1 million adults in the US are hospitalized with pneumonia every year, and about 50,000 die from this disease. Globally, pneumonia is the leading cause of death in children under 5 years old. There are 120 million cases of pneumonia reported each year, and over 10% (14 million) progress to severe episodes. An estimated 935,000 deaths from pneumonia occurred in children under the age of 5 years in 2013.

Figure 1 - Image in a 49-year-old woman with pneumococcal pneumonia.

Chest radiography with posteroanterior and lateral views is the preferred imaging examination for the evaluation of typical bacterial pneumonia (Figure 1). The problem of pneumonia detection based on chest X-Ray Images was already tackled in the literature. (Pranav Rajpurkar, 2017) develops an algorithm that can detect pneumonia from chest X-rays at a level exceeding practicing radiologists. Their algorithm, CheXNet, is a 121-layer convolutional neural network trained on ChestX-ray14, currently the largest publicly available chest X-ray dataset, containing over 100,000 frontal-view X-ray images with 14 diseases. (Parveen NR, 2011) used unsupervised fuzzy c-means classification learning algorithm for the detection of pneumonia infection. (Benjamin Antin, 2017) also followed the approach of CheXNet with a 121-layer dense Convolutional Neural Network (DenseNet).

**Problem Statement**

This project will try to perform a similar task using deep learning techniques. The problem is a binary classification where the inputs are chest X-ray images and the output is one of two classes: pneumonia or non-pneumonia with a certain confidence. Data on chest X-ray images are provided on Kaggle site (Mooney). This dataset includes more than 5000 images split in two categories: normal and pneumonia. In order to be able to provide such solution, the following strategy is applied:

* Implementation of a CNN model build from scratch
* Optimization of the previous CNN model
  + Dropout layers
  + Data augmentation
* Implementation of a CNN model using transfer learning technique
  + Using the bottleneck features of a pre-trained network
  + fine-tuning the first layers of a pre-trained network
* Implementation of a class activation maps for visualizing where deep learning networks pay attention

The problem comes down to being able to determine if lungs are infected with pneumonia or not and where is located the infected area.

**Metrics**

The proposed project is based on a two-stage classification of lung analysis: Normal or Pneumonia. By paying attention to the provided dataset on chest X-ray images, it can be seen that the two classes are not well balanced. The accuracy evaluation metric is discarded in order to avoid the accuracy paradox. Therefore, the use of F1 score as a measure of metric evaluation is more accurate than the accuracy measure. F1 score ranges between 0 and 1. It is defined as:

Table 1 - Confusion Matrix



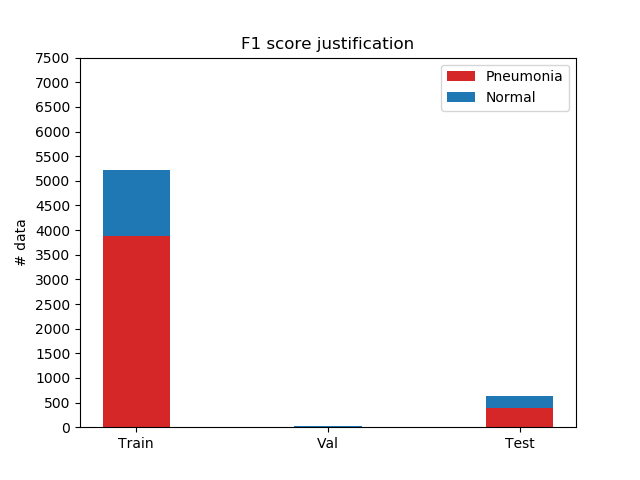
**II. Analysis**

**Data Exploration**

The dataset used for this project was provided on Kaggle site and upload by Paul Mooney[[1]](#footnote-1). This set is already split in three folders and each one is subdivided in two classes:

* Train file (5218 images)
  + Normal (± 25%)
  + Pneumonia (± 75%)
* Val file (9 images)
  + Normal (50%)
  + Pneumonia (50%)
* Test file (624 images)
  + Normal (± 37%)
  + Pneumonia (±63%)

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. F1 is usually more useful than accuracy, especially if you have an uneven class distribution which is true by observing Graph 1:

****

Graph 1 - F1 score justification

All of the images includes in the dataset are in a gray scale format but with various size. This first analysis, only based on the number of images, the size of the images and the ratio between each classes, gives a clue on which techniques will be required (data augmentation, F1 score and standardization method).

**Exploratory Visualization**

In contrast to some datasets, the visualization and extraction of features and properties is quite complex in this case. Only a trained eye, such as a radiologist, is able to recognize and discern any differences between the two images below. On the left image (*Figure 2*), it is the lungs chest X-ray of a patient in good health (not being infected with pneumonia). On the right image (Figure 3), this is the chest X-ray of a patient with pneumonia.

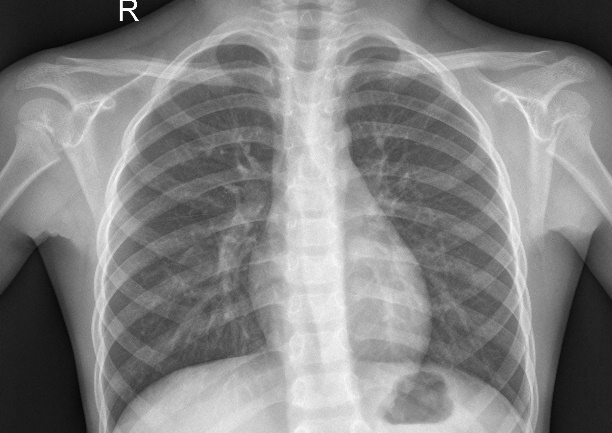
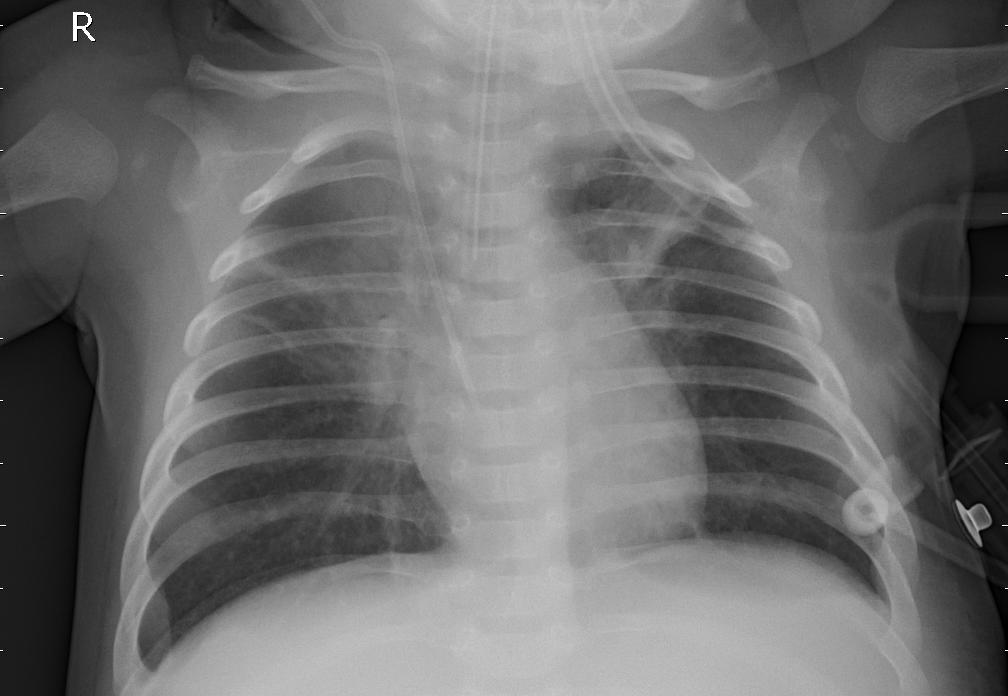
 

Figure 2 - Patient not infected by pneumonia Figure 3 - Patient infected by pneumonia

By analyzing the two images, we understand the importance of implementing deep learning in the medical field. These neural networks are able to detect and recognize patterns that humans cannot detect. Moreover, by showing the potential infected areas, doctors can improve their accuracy. Decision support is therefore increased.

**Algorithms and Techniques**

In order to solve the problem, the following algorithms and techniques are followed. As a starting point, a vanilla CNN was built from scratch in a completely random way. The schematic diagram is drawn in Figure 4. The following parameters for the optimizer were selected:

Table 2 - Parameters for the vanilla model

|  |  |  |
| --- | --- | --- |
| Optimizer | Loss | Metrics |
| rmsprop | Binary cross-entropy | F1 |

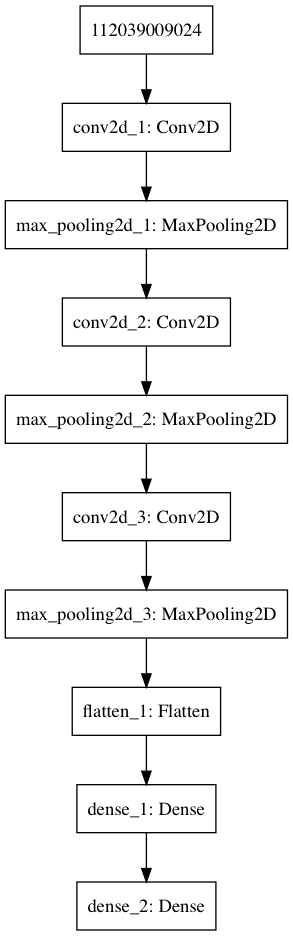


Figure 4 - Schematic diagram of the vanilla CNN

In order to improve the performance obtained with the previous CNN, some dropout layers were added (Figure 5) and others optimizer were tested. Moreover, a number of random transformations (ImageDataGenerator) were applied on the validation set in order to increase the number of data. This helps prevent overfitting and helps the model generalize better.

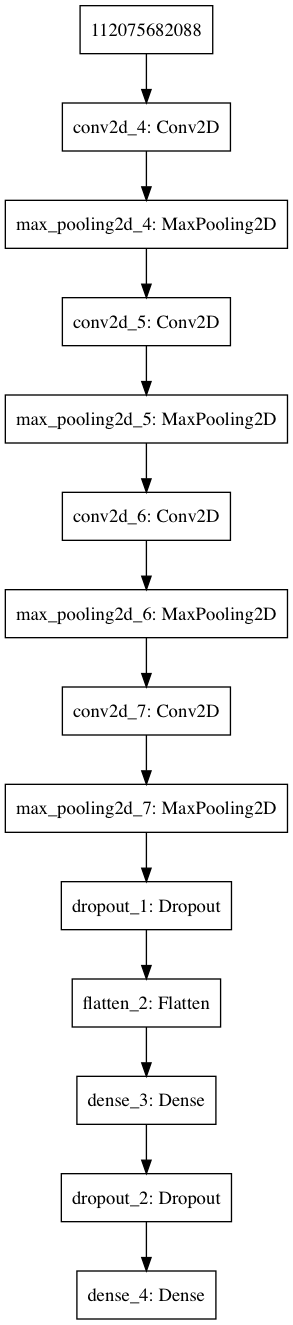


Figure 5 - Schematic diagram of the optimized CNN

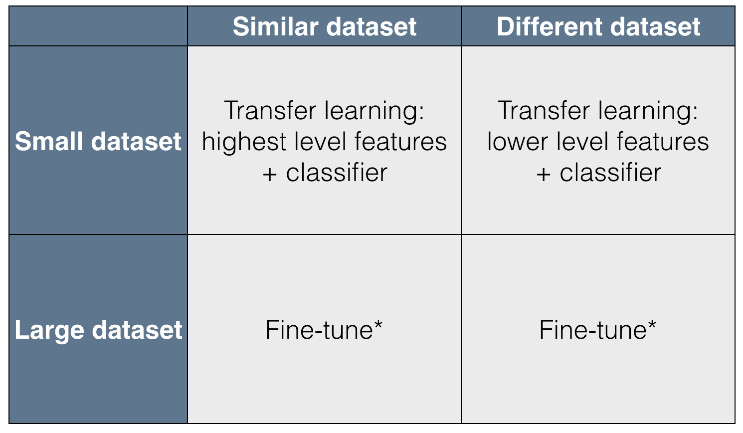
The following parameters for the optimizer were selected:

Table 3 - Compilation parameters

|  |  |  |
| --- | --- | --- |
| Optimizer | Loss | Metrics |
| Adam | Binary cross-entropy | F1 |
| SGD | Binary cross-entropy | F1 |

After that, in this project, the pre-trained Inception-V3 model was selected (Figure 6). Inception-v3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012. This is a standard task in computer vision, where models try to classify entire images into 1000 classes, like "Zebra", "Dalmatian", and "Dishwasher". Because the dataset of this project is completely different from Inception-V3 one, the bottleneck features of a pre-trained network is used and then the fine-tuning of the first layers was performed as suggested in Table 4.

Table 4 - Transfer learning justification



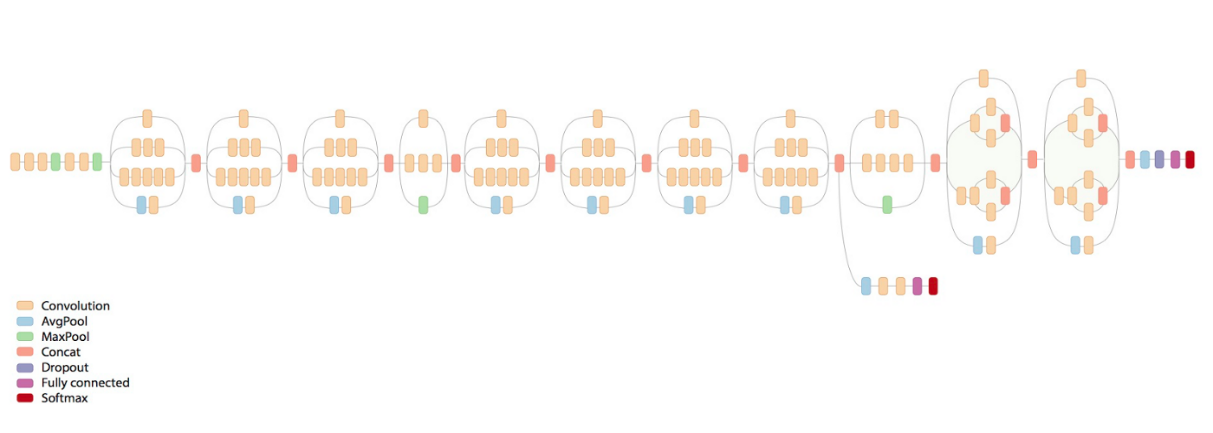


Figure 6 - Schematic diagram of Inception V3

Finally, a Class Activation Maps was implemented to get the discriminative image regions used by a CNN to identify a specific class in the image. A class activation map (CAM) let see which regions in the image were relevant to this class.

**Benchmark**

By definition the benchmark acts as a threshold that can determine if the project has succeeded or not. The F1 value returned with the vanilla CNN model will be used as a reference. Any performance of improvement will lead to a higher score than the benchmark model.

**III. Methodology**

**Data Preprocessing**

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

* *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?*
* *Based on the****Data Exploration****section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?*
* *If no preprocessing is needed, has it been made clear why?*

**Implementation**

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

* *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
* *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
* *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

*F1 not supported by Keras + lien Github (Mise en reference)[[2]](#footnote-2)*

*Explication of the code*

*Utilisation CPU and not GPU*

**Refinement**

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

* *Has an initial solution been found and clearly reported?*
* *Is the process of improvement clearly documented, such as what techniques were used?*
* *Are intermediate and final solutions clearly reported as the process is improved?*

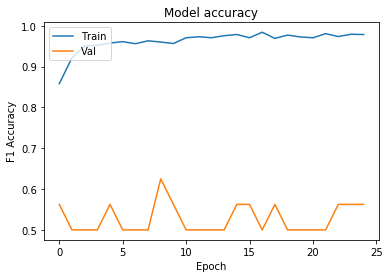
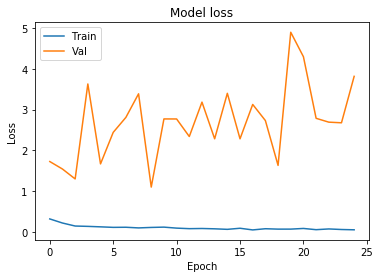
**IV. Results**

**Model Evaluation and Validation**

During development, a validation set was used to evaluate the model. The final architecture and hyperparameters were chosen because they performed the best among the tried combinations. Here is a list of every parameters used in optimized CNN architecture model :

* The shape of the filters of the convolutional layers is 2\*2.
* The first convolutional layer learns 16 filters, the second learns 32 filters, the third one learns 64 filters and the fourth one learns 16 filters.
* The paddings were defined as same.
* The convolutional layers have a stride of 2, so the resolution of the output matrices is half the resolution of the input matrices.
* The pooling layers have a size of 2. The last pooling layers has a size of 4.
* The first fully connected layer has 500 outputs, the second 2. (The outputs of the latter correspond to the two classes, “Normal” and “Pneumonia”).
* Two dropout were needed (0.2 and 0.4) in order to reduce the overfitting parameters.
* The Relu activation function was selected on intermediate layers. For the last one layer, the softmax function was implemented.

Graph 2 and 3 shows that even on a small number of epochs, the F1 score quickly reach out a plateau for the train set and the loss tends towards an asymptotic value.

Graph 2 - F1 accuracy evolution Graph 3 - Loss evolution

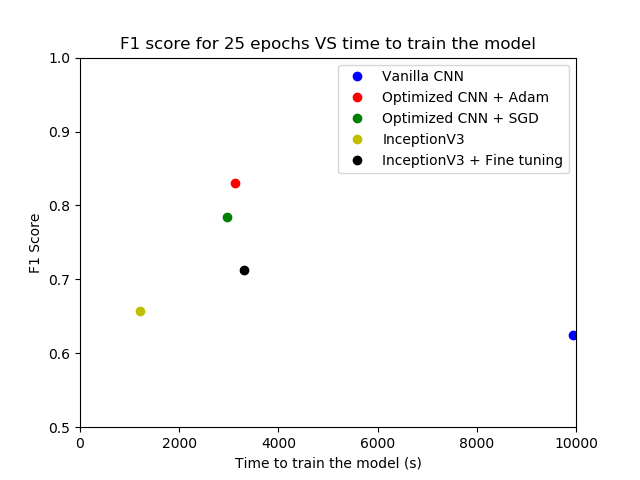
To verify the robustness of the final model, a test was conducted by implementing a class activation maps on a test sample (Figure 7 and 8). Even if it needs to be confirmed by a specialist, the heat map highlights some region that may indeed be infected by the pneumonia.

**Justification**

The model’s solution and its results are compared to the benchmark previously established on Table 5 and Graph 4. The F1 score value significantly increased in the case of the optimized CNN model architecture with the adam optimizer. The computation time is also quiet acceptable knowing that only a CPU (Intel HD Graphics 5000 1536 Mo) was used. It is also interesting to note that the transfer learning combined with fine-tuning is a powerful technique. By only train the first layers and freeze the rest of them, the F1 score increases from 0.657 to 0.713.

Table 5 - F1 scores

|  |  |
| --- | --- |
|  | F1 score on test set |
| Vanilla CNN with adam optimizer | 0.625 |
| Optimized CNN with adam optimizer | 0.831 |
| Optimized CNN with SGD optimizer | 0.785 |
| Inception-V3 pre-trained model | 0.657 |
| Inception-V3 pre-trained model + fine tuning | 0.713 |

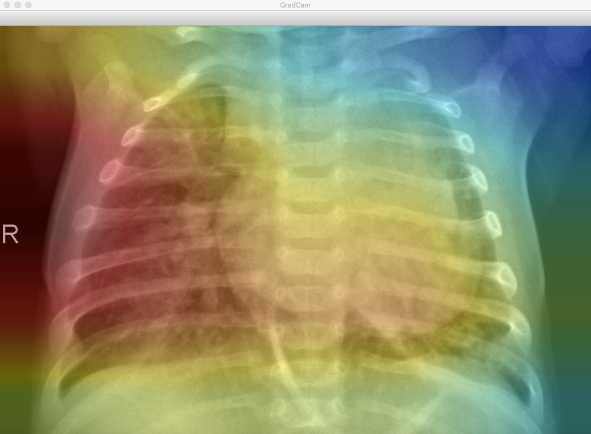


Graph 4 - F1 score vs computation time

**V. Conclusion**

**Free-Form Visualization**

As mentioned earlier, the visualization and extraction of patterns is crucial in this project. Having an algorithm that can predict with some confidence the probability of being infected is crucial but it becomes even more efficient if the algorithm is able to show and to determine the suspected area of the disease. That's why the implementation of a class activation maps for visualizing where deep learning networks pay attention is a powerful tool. On the left image (*Figure 7*), it is the chest X-ray of a patient infected by the pneumonia. On the right image (Figure 8), the heat map highlights some region of interest detected by the algorithms. That can be a good starting for doctors to know where to look at.



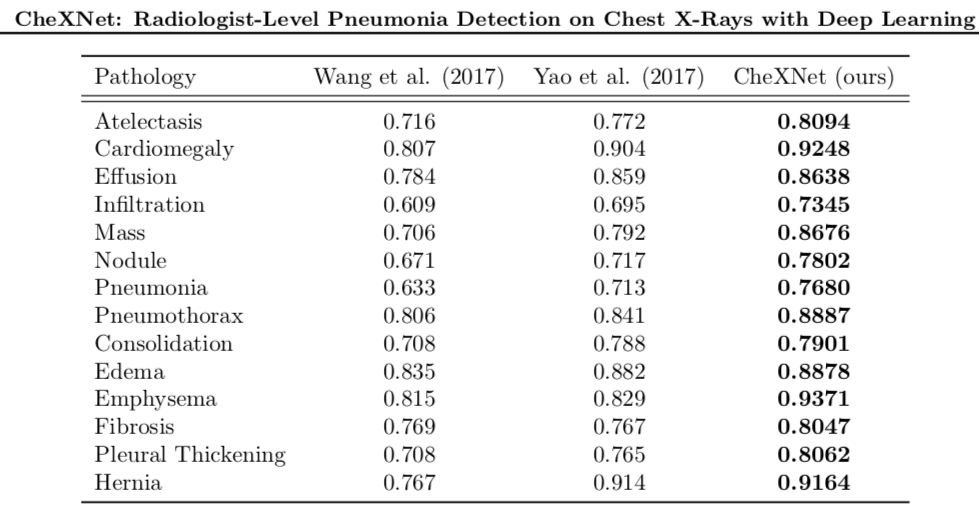
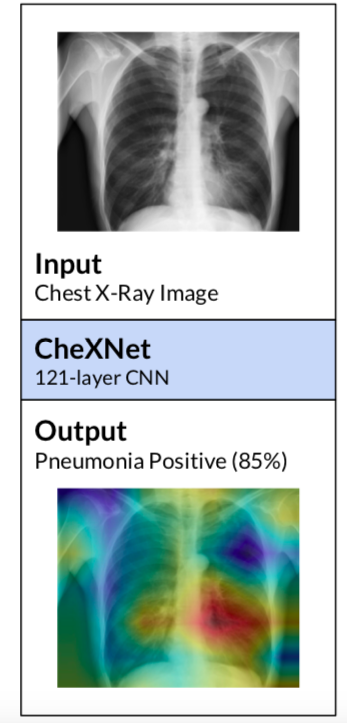
*Figure 7 - Infected lungs* Figure 8 - class activation maps predictions

**Reflection**

The goal of this project was to find a way to correctly and with some accuracy recognize lungs infected by the pneumonia. To achieve this, various CNN model architectures were tested. Starting from a vanilla model, the accuracy was improved by using some dropout layers to the original model. Techniques such as data augmentation, tuning on hyperparameters and selection on various optimizers were tested as well. The objective was to obtain the highest evaluation score as possible. Because of the non-balanced classes issue, the F1 score was selected. Finally, a good compromise between the computed time and the F1 score was found.

**Improvement**

The problem itself is quite simple and binary. It is also very easy to imagine that the model could be improved by using more available data. Choosing to train the program on a GPU instead of a CPU could drastically decrease the computation time as well allowing more complex CNN architecture. The last layers could also be split in more than just two classes to detect others lung disease. As an example, Standford researcher created a 121-layer convolutional neural network (CheXNet) trained on ChestX-ray14, currently the largest publicly available chest X- ray dataset, containing over 100,000 frontal-view X-ray images with 14 diseases. As provided below, they get an accuracy 76% for detecting pneumonia in infected lungs.



**VI. Bibliography**

**Benjamin Antin Joshua Kravitz, Emil Martayan** Detecting Pneumonia in Chest X-Rays with Supervised [Revue]. - 2017.

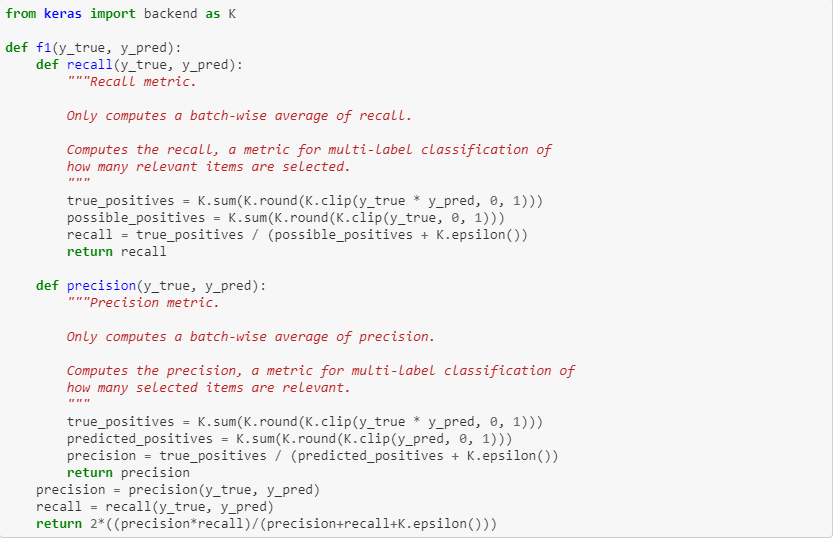
**Mooney Paul** [En ligne]. - https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

**Parveen NR Sathik MM** Detection of Pneumonia in chest X-ray images. [Revue]. - 2011.

**Pranav Rajpurkar Jeremy Irvin, Kaylie Zhu, and Co** CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays [Revue]. - 2017.

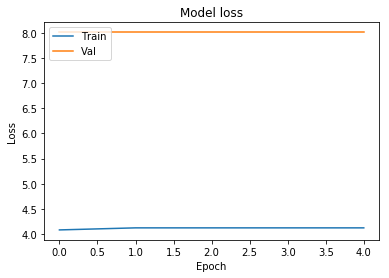
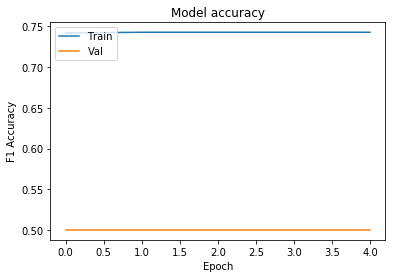
**VII. Annex**

**F1 score implementation**

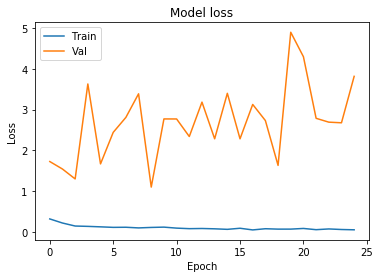
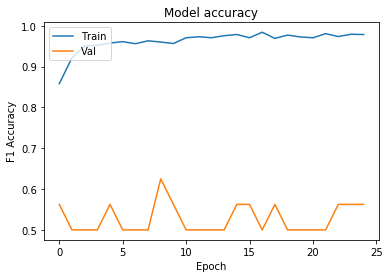


**Plot of the training and validation losses**

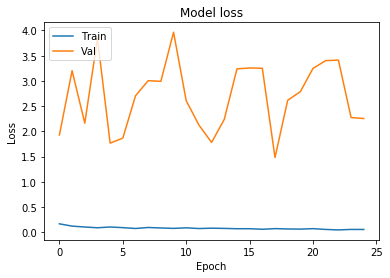
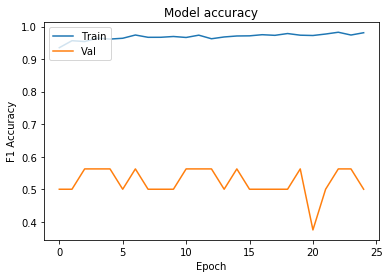
Vanilla CNN with adam optimizer



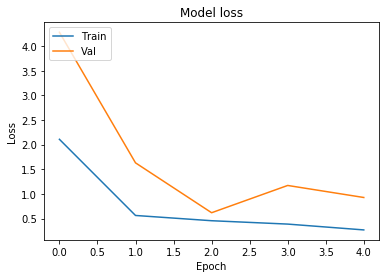
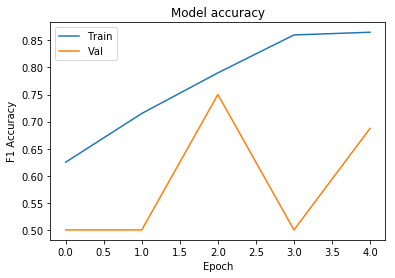
Optimized CNN with adam optimizer



Optimized CNN with SGD optimizer



Inception-V3 pre-trained model



1. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/downloads/chest-xray-pneumonia.zip/2 [↑](#footnote-ref-1)
2. https://github.com/FrancoisMasson1990/Machine\_Learming\_Engineer\_Nanodegree\_Program/tree/master/Capstone\_Final\_Francois [↑](#footnote-ref-2)