Université de Paris

Kaggle Project

Detection of pneumonia in chest X-ray images using a deep learning approach

AUCOUTURIER Camille
BALOCHE Valentin

Reference:
Daniel S. Kermany et al. *Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning*. Cell, Feb 22; 175(2): 112-1131.

GitHub repository:
https://github.com/AUCAM/KAGGLE_Project

M2BI Learning Methods
2021/2022

Foreword

Pneumonia is an infection that causes inflammation in one or both of the lungs and may be provoked by virus, bacteria, fungi or other germs. It can affect people of any age, but the greatest risk is in young children, the elderly, and immunocompromised patients. It can sometimes lead to serious complications so it is important to seek immediate diagnosis and medical attention.

When pneumonia is suspected, a chest X-ray is usually performed to confirm the diagnosis. The interpretation is based on the observation of white spots in the lung that identify the infection. Depending on the cause of the infection (virus or bacteria), the patterns can differ from a single white condensed area (bacterial pneumonia) to a more diffuse one (viral pneumonia). Overall, this analysis is not always easy to perform because of the initial stages of the disease, or because it involves a part of the lung not easily seen by X-ray.

The goal of this kaggle project is to create a convolutional neural network (CNN) which is able to detect pneumonia from chest X-ray. It is a part of a more global framework aimed at developing algorithms demonstrating performance comparable to that of human experts.

In the first part of this report we will present a first version of a CNN that can respond to the problem. We will then present the process we have been through to improve its efficiency.

Data

Chest-X-Ray images were selected from retrospective cohorts of pediatric patients of 1 to 5 years old from Guangzhou Women and Children's Medical Center. The dataset is composed of 5856 X-ray colored pictures (JPEG) whose size may vary from one image to another, but with an average size of 1500 x 1700 pixels. We have made the choice to extract 20% of the initial train data to create our validation set (*Fig. 1*).

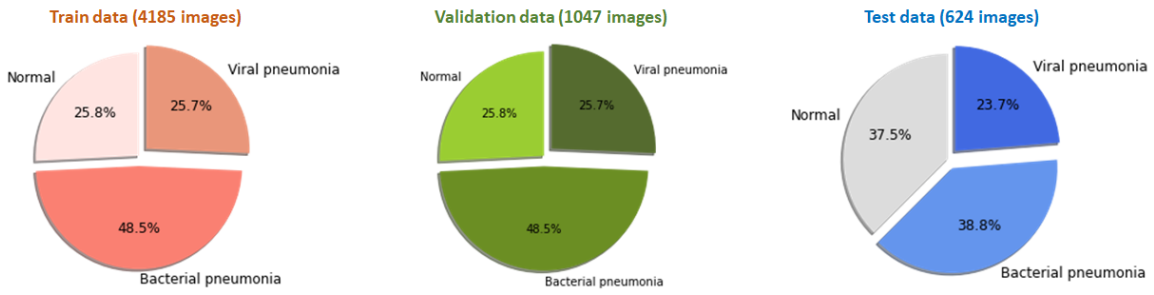


Figure 1: Data repartition for train, validation and test sets

Network development

I- Loading and preprocessing data

In order to reduce and homogenize the images size, we started by reshaping them to a 224 x 224 pixels size using “load_img” from the package tensorflow.keras.preprocessing.image which uses a nearest neighbor interpolation method. We kept the 3 color channels to be able to reuse models pretrained with ImageNet, a large database of colored pictures.

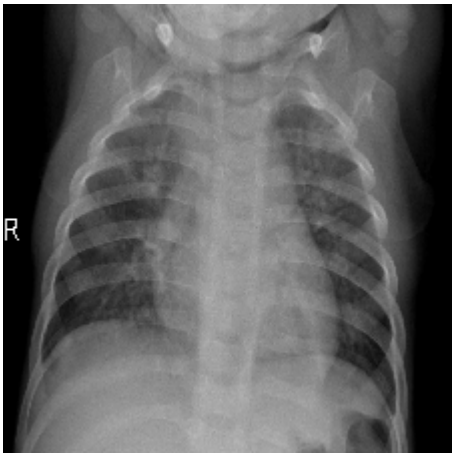


Figure 2: Resized image to 224 x 224 from the bacterial pneumonia data

For the choice of the size, we studied what was done in other projects, whose image sizes globally range between 150 and 224. Having in mind the project to make the classification more complex (from binary normal /pneumonia to a 3rd class normal/viral pneumonia/bacterial pneumonia), we chose to keep the best resolution (*Fig. 2*). After having converted images in arrays, we finally scaled them to the [0, 1] range in order to normalize this data.

II- Building a 2 classes CNN model

We created the two class vectors corresponding to the categories normal/pneumonia and one hot encoded them.

For the construction of our network, we first tried to implement some data augmentation in our images in order to avoid over-fitting. After several trials with random zooms, contrasts or flips, we finally settled on a single random horizontal flip step, which gave us the best results.

We built a model of 3 successive convolution layers with a progressive increase of the filters (32 -> 64 -> 128) with a kernel size (3x3) that moves in stride of 1, to extract the different features of the images. Each convolution layer was followed by Max Pooling. Then, those layers were flatten and we added a dropout and a last dense layer for the classification (normal or pneumonia). Overall, this model was based on 266,306 trainable parameters (*Fig. 3*).

Concerning the choice of hyperparameters, we initially started with a standard batch size of 32 and used Nadam optimizer with default parameters. At this point, we had some problems with over-fitting, characterized by an improvement of the accuracy but with fluctuating validations. We then switched to Adam optimizer with a learning rate of 0.0001 and a learning rate decay of 1e-5, which greatly solved our issue.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
random_flip (RandomFlip)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 26, 26, 128)	0
flatten (Flatten)	(None, 86528)	0
dropout (Dropout)	(None, 86528)	0
dense (Dense)	(None, 2)	173058

Total params: 266,306
Trainable params: 266,306
Non-trainable params: 0

Figure 3: Model summary (CNN_1 - 2 classes)

We chose to shuffle the train data because pneumonia files were ordered based on the infection origin (bacterial or viral) and we thought that could condition the network training. That is also why we used our own validation data instead of the validation.split option which picks the last data of a file to automatically generate the validation set.

Finally, we compiled the model using a categorical cross entropy loss function, with the idea of subsequently transposing our network to a 3 classes prediction model. The training process stopped at 8 epochs when the validation loss reached a plateau (*Fig. 4*).

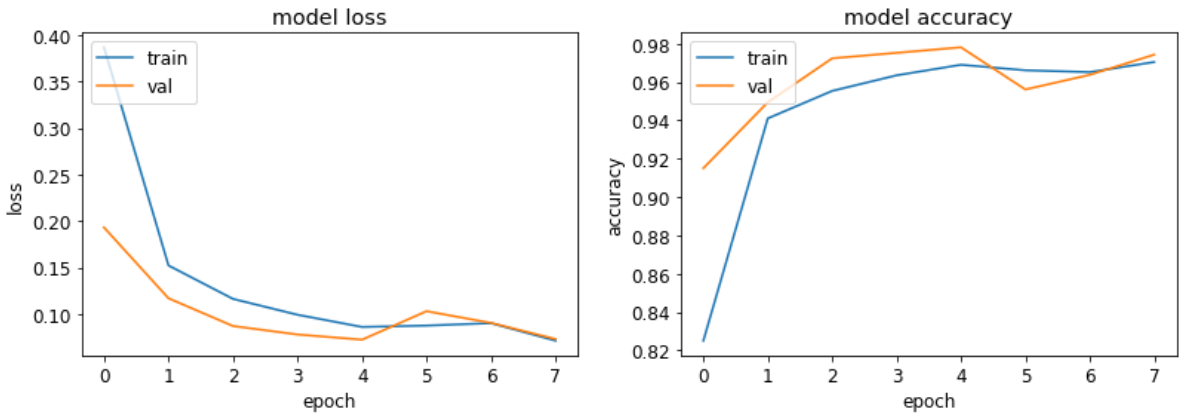


Figure 4: Plot history of loss/accuracy (CNN_1 - 2 classes)

We launched several tests to verify the effectiveness and consistency of the results we obtained. In all cases, we ended up with an accuracy close to 82% with our test data (*Fig. 5*).

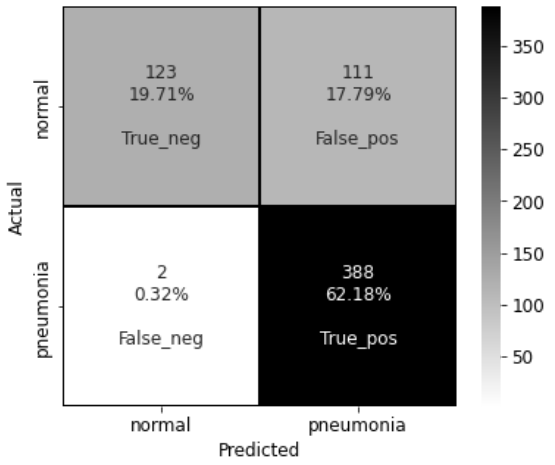


Figure 5: Confusion matrix (CNN_1 - 2 classes)

Interestingly, when we looked more closely at the predictions, we saw that the apparent good accuracy was due to a bias in the data distribution. Indeed, while the CNN had a very good sensitivity for detecting pneumonia (388/390), it was however not very specific and also categorized a lot of normal images in the pneumonia category (111/234). In a way, the pneumonia images representing 63% of the dataset, we "favored" the efficiency of our network by giving it data that it sorts efficiently.

It was not relevant to us to further improve the accuracy of our 2 classes CNN model. Instead, we tried to apply its architecture to a 3 classes CNN model to see how it would react.

III- Building a 3 classes CNN model

This attempt to classify pneumonia into subgroups depending on the infection origin is not the most popular when we look at the other kaggle projects. Indeed, the first task of this project was to detect pneumonia from chest-X-ray. However, we thought this could be interesting to make the analysis more complex, knowing that differences can be observed by human experts.

a) Transposition

To fit with the three classes prediction, we created the three class vectors normal/bacterial_pneumonia/viral_pneumonia the same way we did before, and one hot encoded them. After modifying the expected number of classes, we trained the model. The addition of an output led to an increase of trainable parameters to a final value of 352, 835.

After running some training tests, we chose to keep the same hyperparameters and simply increased the batch size from 32 to 64 in order to generalize the training and limit the fluctuations of the validation curve.

The training stopped after 8 epochs (*Fig. 6*) and the evaluation of the model gave us 71% of accuracy.

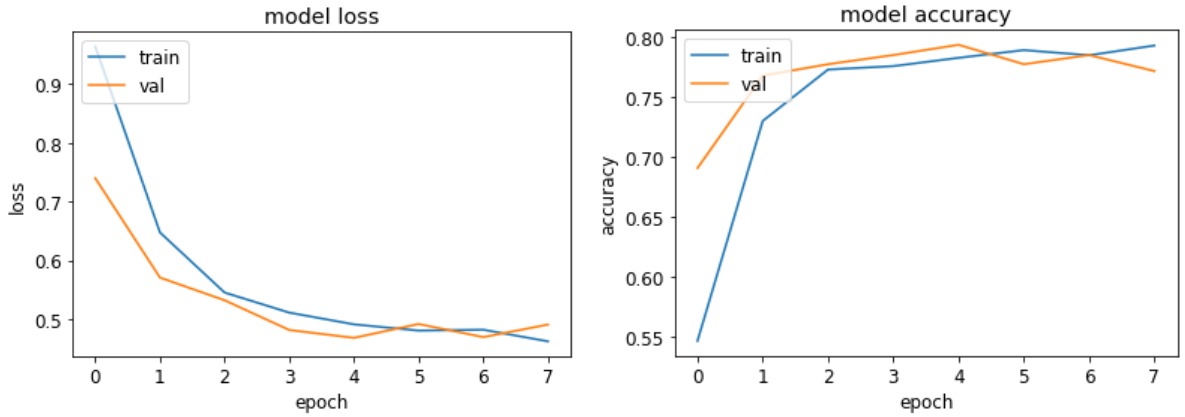


Figure 6: Plot history of loss/accuracy (CNN_1 - 3 classes)

Although the efficiency of the network for categorizing in 3 classes was lower than for categorizing in 2, the errors made in both cases were not the same. Indeed, while with 2 classes the network had difficulty to predict the normal condition, with 3 classes the normal category was, on the contrary, the most accurately predicted (*Fig. 7*).

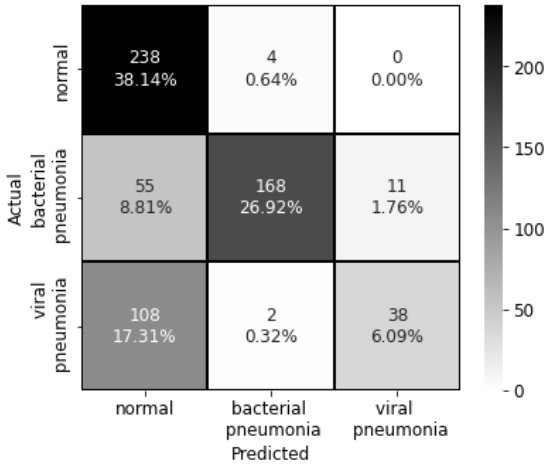


Figure 7: Confusion matrix (CNN_1 - 3 classes)

Our hypothesis is that in the 2 classes prediction, several images of normal chests were categorized as pneumonia because of similarities between normal and viral pneumonia images. This is indeed something that can be observed in the confusion matrix of Fig. 7. This hypothesis is further fueled by the fact that viral pneumonia are characterized by discrete and diffuse opacities among the lungs, while bacterial pneumonia are characterized by visible focal opacities making them easier to differentiate from normal lungs.

In order to create a more efficient prediction model, we thought that the use of a pre-trained model could allow us to detect more subtle differences.

b) Transfer learning using Inception V3

We chose to use the Inception V3 network, which is a convolutional neural network from the Inception family which include several improvements like Label Smoothing, Factorized 7 x 7 convolutions, or the use of an auxiliary classifier to propagate label information lower down the network (along with the use of batch normalization for layers in the side hear) (*Annex 1*).

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
random_flip_1 (RandomFlip)	(None, 224, 224, 3)	0
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
flatten_1 (Flatten)	(None, 51200)	0
dense_1 (Dense)	(None, 256)	13107456
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 3)	387

Total params: 34,943,523
Trainable params: 13,140,739
Non-trainable params: 21,802,784

Figure 8: Model summary (transfer learning using Inception V3 - 3 classes)

So we used this model, already trained on the huge ImageNet database which is composed of 14,197,122 annotated images. We freezed all pretrained weights, and trained only some new layers to fit with our chest X-rays images classification. We added two dense layers, each followed by a dropout, for a total of 34,943,523 parameters including 13,140,739 trainable (*Fig. 8*)

Despite the large number of trainable parameters, the training stopped at 10 epochs (Fig. 9), reaching an accuracy of 75% on test data (Fig. 10). This early training stop was due to the constancy of the validation curves which didn't benefit from the learning carried out by the network. By increasing the number of epochs, we would have observed an over-fitting phenomenon.

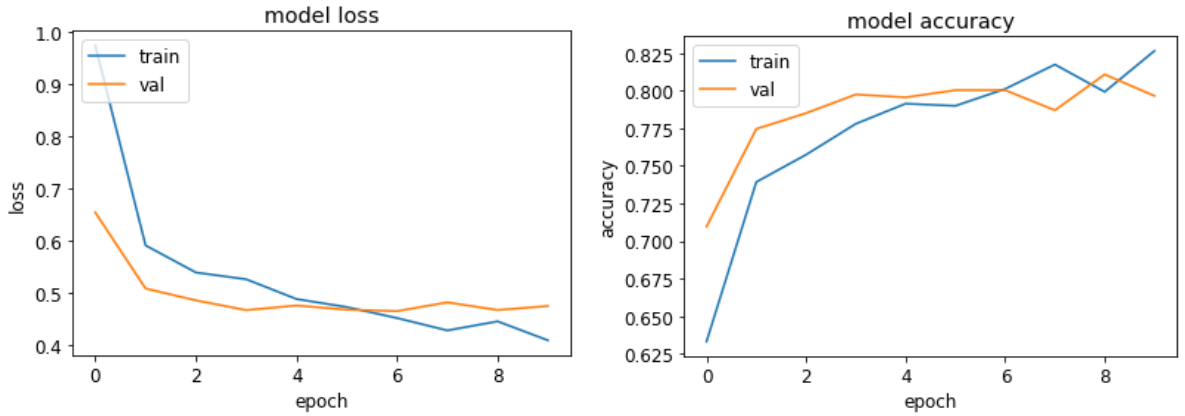


Figure 9: Plot history of loss/accuracy (transfer learning using Inception V3 - 3 classes)

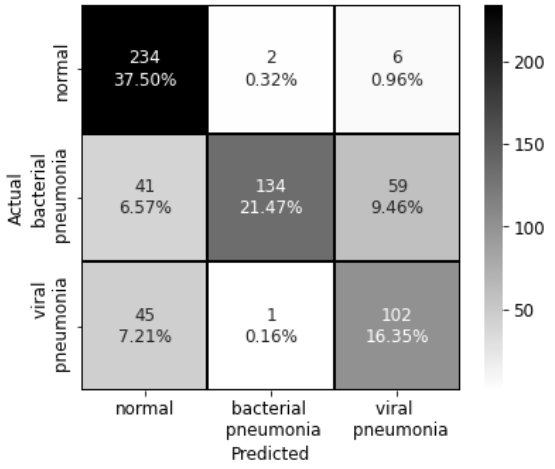


Figure 10: Confusion matrix (transfer learning using Inception V3 - 3 classes)

We tried to solve this problem in several ways, by modifying the network architecture or by adjusting the hyperparameters, but to no avail. It seemed to us that this network, trained with a great diversity of images, was perhaps not adapted to the analysis of our very similar images. We then thought that it would be more relevant to exploit transfer learning using the first network we built, which was able to effectively recognize normal images.

c) Transfer learning using CNN_1

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 224, 224, 3)]	0
model_2 (Functional)	(None, 26, 26, 128)	93248
conv2d_97 (Conv2D)	(None, 24, 24, 32)	36896
max_pooling2d_7 (MaxPooling2	(None, 12, 12, 32)	0
conv2d_98 (Conv2D)	(None, 10, 10, 64)	18496
max_pooling2d_8 (MaxPooling2	(None, 5, 5, 64)	0
conv2d_99 (Conv2D)	(None, 3, 3, 128)	73856
max_pooling2d_9 (MaxPooling2	(None, 1, 1, 128)	0
flatten_3 (Flatten)	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 3)	387

Total params: 222,883
Trainable params: 129,635
Non-trainable params: 93,248

Figure 11: Model summary (transfer learning using CNN_1 - 3 classes)

As previously explained, the first network we developed (CNN_1), applied on a 3 classes prediction, succeeded in identifying normal images. We therefore used this pre-trained network, removed the last layers (flatten, dropout and dense layers), fixed the weights of the remaining ones and added at the end a new network with the exact same architecture as the CNN_1 (*Fig. 11*).

While keeping the same hyperparameters, we launched a training which lasted much longer than what we get with the other networks (29 epochs) (*Fig. 12*). It finally allowed us to markedly improve the accuracy of our prediction on the test data (84%)(*Fig. 13*).

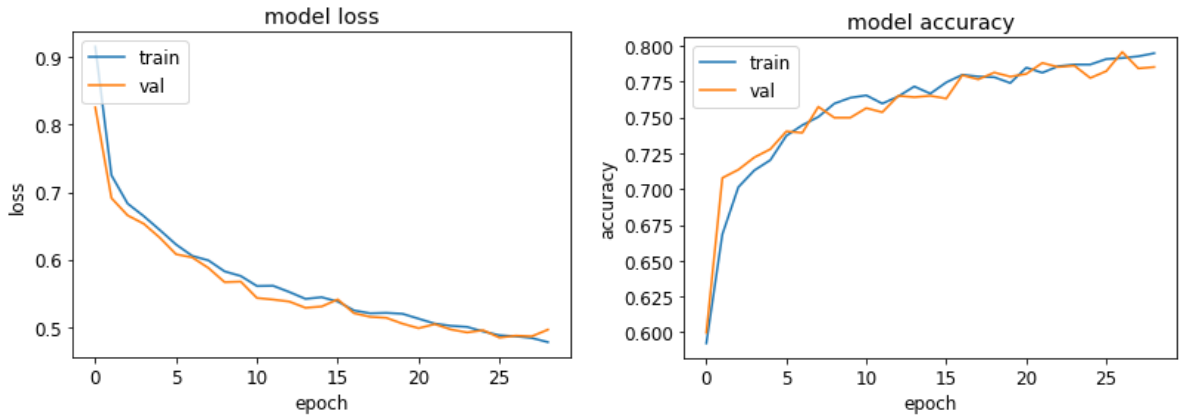


Figure 12: Plot history of loss/accuracy (transfer learning using CNN_1 - 3 classes)

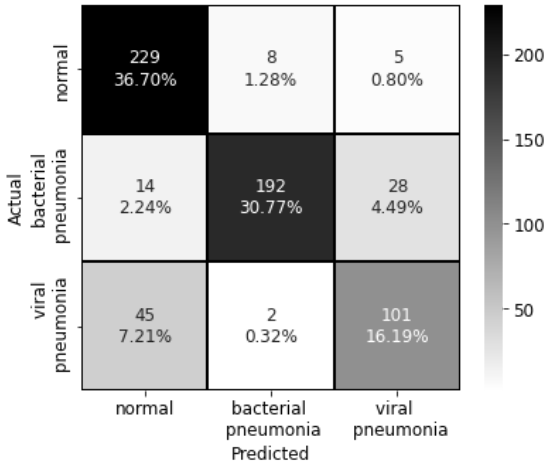


Figure 13: Confusion matrix (transfer learning using CNN_1 - 3 classes)

The confusion matrix visible in Fig. 13 allowed us to realize that this model retained its capacity to recognize normal images, but also and mostly, considerably increased its capacity to recognize viral pneumonia. Overall, these results highlight the great potential of transfer learning, which can make possible the "save" of learning states, in order to allow a more tuned and progressive learning.

Conclusion

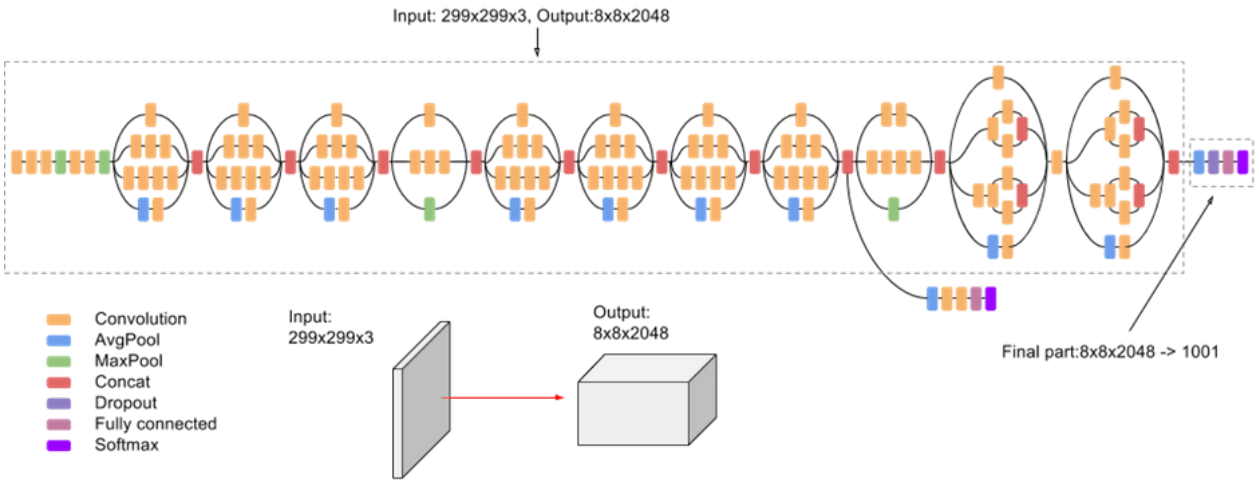
In conclusion, we succeeded in creating a CNN capable of classifying chest-X-ray images based on their normal vs. pneumonia status. Moreover, among the pneumonia chest images, our network was able, for the major part of them, to identify those of viral origin and those of bacterial origin.

We could of course have explored the efficiency of many other networks, for example by carrying out transfer learning with other CNN (ex: AlexNet, DenseNet, …), with some of them pre-trained on other datasets (ex: chest-X-ray from diverse pathologies), but that would have gone beyond the research framework of this project.

Nevertheless, looking at the plot history of our last model, the good follow-up of the validation curve along the training curve lets us think that the efficiency could have been slightly improved by increasing the patience of the callback, and therefore the number of epochs. However, we would certainly have faced material limitations. Indeed, we used the services offered by Google Colab in order to carry out this project. It has the advantage of offering a functional and easy to use environment but finds its limits in the size and duration of the allocated resources.

Speaking of the limitations we encountered, several of them concerned over-fitting issues. Most of them could be solved by finely modifying the hyperparameters, highlighting the importance of those, maybe beyond the network's architecture itself.

Annex



Annex 1: Inception V3 architecture