



Rapport de projet :

Analyse de radiographies pulmonaires Covid-19

Data Scientist - Mars 2024 (Bootcamp)

Equipe projet :

Thomas Baret,
Nicolas Bouzinbi
Florent Daydé
Nicolas Fenassile

Tuteur projet :

Gaël Pennessot

Table des matières

1. Introduction.....	3
1.1. Contexte.....	3
1.2. Problématique.....	3
2. Description des données.....	4
2.1. Contenu des jeux de données.....	5
2.2. Intégrité des données.....	5
2.3. Effectifs par type d'infection.....	10
2.4. Intensités de pixels moyennes.....	14
2.5. Images moyennes.....	17
2.6. Conclusion sur le choix du jeu de données.....	18
3. Preprocessing des images.....	19
3.1. Normalisation Min-Max.....	19
3.2. Normalisation par égalisation d'histogrammes.....	20
3.3. Application des masques sur les images.....	21
4. Première approche du deep-learning.....	23
4.1. Fonctionnement d'un modèle de deep learning.....	23
4.2. Application à notre jeu de données.....	23
4.3. Modèle LeNet.....	24
5. Transfer Learning.....	27
5.1. Modèle Xception.....	28
5.2. Modèle DenseNet201.....	33
5.3. MobileNetV2 [22].....	38
5.4. VGG19.....	47
5.5. Conclusion sur les modèles.....	54
6. Conclusion.....	56
7. Bibliographie.....	58
8. Environnement de travail.....	59
8.1. Ressources.....	59
8.2. Librairies Python.....	60

1. Introduction

1.1. Contexte

La pandémie de COVID-19 a été un véritable défi de santé publique mondial, forçant les autorités sanitaires à s'adapter le plus vite possible pour l'enrayer. En l'absence de traitement et de vaccin, la prévention était l'unique pilier sanitaire sur lequel il était possible d'agir. Ainsi, un moyen de dépistage fiable et rapide est devenu indispensable dans la lutte contre la pandémie. Dans un premier temps, deux tests moléculaires ont été mis en place pour le dépistage [1] : un test antigénique basé sur l'hybridation des protéines et un test PCR (Polymerase Chain Reaction). Cependant, ces tests rencontrent plusieurs problèmes : le test antigénique, même si rapide (15 minutes), présente un fort taux de faux-négatifs, donc un problème de sensibilité [2] ; le test PCR, quant à lui, est long à réaliser, et bien qu'étant plus fiable, peut présenter une sensibilité entre 50 et 62% [3] : plusieurs tests doivent être effectués sur une période de 14 jours pour confirmer les résultats.

La COVID-19 étant un virus pulmonaire, les méthodes d'imageries médicales comme les radios ou scanners peuvent être aussi utilisées pour le diagnostic et la gestion de la maladie. De plus, l'état des poumons ainsi que l'avancement de la maladie peuvent également être estimés par les radiologues : en effet, il existe certaines lésions caractéristiques dues à la COVID-19 [4]. En outre, une radio pulmonaire ne prend en moyenne que 15 minutes.

Avec l'avènement du Deep Learning (DL), le domaine de l'imagerie médicale s'est trouvé en première ligne dans l'application de modèles permettant une meilleure stratégie de diagnostic et d'analyse pour un grand nombre d'affections. Par exemple, grâce à l'utilisation de CNNs (Convolutional Neural Networks) a pu être appliqué avec succès sur des images microscopiques, la classification de tumeurs cérébrales, des images IRM et des photos de la rétine [5], [6], [7], [8], [9] . Ces modèles permettent de diagnostiquer et d'analyser presque instantanément une image qui leur est fournie, avec une très grande précision (parfois meilleure que celle des radiologues).

L'utilisation de modèles de Deep Learning pour la détection rapide de la COVID-19, et éventuellement l'analyse de la progression de celle-ci était donc une étape logique dans la lutte contre cette maladie, autant pour diminuer la progression de la pandémie que pour une prise en charge rapide des patients avec une meilleure qualité de soins.

1.2. Problématique

En se basant sur l'article ayant servi de fondement au projet [10] , nous avons décidé de construire un modèle de deep learning capable de différencier de façon précise entre une radio d'un patient COVID, Non-COVID et Normal. **Nous nous retrouvons donc face à un problème de classification multiple (3 catégories)**. Dans cet article, les auteurs ont également mis en place un modèle capable de segmenter les poumons sur les radios, afin de concentrer la recherche de features pour détecter l'infection. Nous avons choisi dans un premier temps de travailler sur un modèle fiable capable de classer correctement les radios sans masque, et ensuite de tester ce modèle sur les mêmes images après application des masques correspondants. Nous avons donc chacun choisi au moins deux modèles à tester et à comparer. Le modèle doit être capable de reconnaître visuellement des features

permettant de différencier différents types d'images, il s'agit donc d'une tâche de reconnaissance.

Après une revue exhaustive de la littérature, nous avons sélectionné les modèles en nous basant sur deux critères : d'abord un ensemble de métriques (accuracy, recall et f1-score) et ensuite la faisabilité de la mise en place du modèle, c'est à dire le temps de tuning, d'entraînement et de fine tuning avec les moyens dont nous disposons pour les faire tourner : kaggle, google colab et ordinateur personnel.

La métrique de performance principale utilisée pour comparer les modèles est la précision (accuracy). Le choix de cette métrique s'est fait naturellement puisque ayant à faire un problème de classification avec un grand nombre de données, réparties en classes équilibrées, il s'agit d'une métrique simple et fiable pour comparer les modèles dans un premier temps.

Notre but était donc de trouver un modèle fiable et relativement rapide à mettre en place, que nous pourrions améliorer pour pouvoir classifier avec un précision d'au moins 90 % des images de radiographie pulmonaire.

2. Description des données

Les données sont issues du projet Kaggle “COVID-19 Radiography Database” :
<https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database/data>

Ce jeu de données a été construit initialement en 2020 lors de l'émergence de la Covid-19 par une équipe internationale de chercheurs dans le but de proposer une méthode d'identification d'une infection par Covid-19 par l'analyse de radiographies pulmonaires par des algorithmes de Deep Learning.

L'avantage de cette méthode d'analyse d'imagerie réside dans sa rapidité, son coût réduit et sa facilité d'accès par rapport à la méthode de diagnostic par RT-PCR.

Ce travail a abouti à deux publications : Chowdhury M.E.H, et al., 2020 [11] et Rahman T., et al., 2020 [12].

Ce projet Kaggle donne un lien vers un second jeu de données nommé “COVID-QU-Ex Dataset”, qui à première vue semble plus complet que le précédent.

<https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database/data>

Il a donné lieu à 3 autres publications : Tahir A.M. et al., 2021 [10], Anas M. et al., 2021 [13] et Degerli A., et al., 2021 [14].

Par la suite, nous allons étudier les deux jeux de données afin de déterminer lequel nous semble le plus pertinent à utiliser dans le cadre de notre projet.

2.1. Contenu des jeux de données

2.1.1. COVID-19_Radiography_Dataset

Les données sont composées de 4 jeux de données correspondant à 4 états des patients :

- COVID : Covid-19
- Lung opacity : infection pulmonaire autre que Covid et autre que Pneumonie virale,
- Normal : pas d'infection pulmonaire
- Viral Pneumonia : pneumonie virale

Chaque jeu de données contient un fichier de metadata au format .xlsx et 2 sous dossiers "images" et "masks" qui contiennent respectivement les radiographies pulmonaires et leurs masques au format .png.

Un fichier README.md accompagne les données, il contient notamment les effectifs par source de données pour chacun des 4 sous-jeux de données.

2.1.2. COVID-QU-Ex Dataset

Ces données sont déjà divisées en 3 sets: Train, Test, Val, qui sont utilisés dans la partie transfert learning. Dans chacun de ces sets, les données sont à nouveau divisées en 3 jeux de données correspondant à 3 états des patients :

- COVID : Covid-19
- Non-Covid infection : infection pulmonaire autre que Covid
- Normal : pas d'infection pulmonaire

Là encore, les jeux de données sont divisés en images et masks. Les metadata ne sont pas directement accessibles sur ce set, il y a seulement un fichier COVID-QU-Ex dataset.txt qui décrit les effectifs pour chaque état.

2.2. Intégrité des données

La première phase de l'exploration des données a consisté en la vérification de l'intégrité des données : cohérence et complétude (données manquantes).

2.2.1. COVID-19_Radiography_Dataset

Pour le set "COVID-19_Radiography_Dataset", nous allons donc comparer le fichier de métadata avec les images réellement présentes dans notre jeu de données.

Fichiers de metadata

jeu de donnée	colonne	N observations	N valeurs uniques	N NA	exemple
COVID	FILE NAME	3616	3616	0	COVID-1
COVID	FORMAT	3616	1	0	PNG
COVID	SIZE	3616	1	0	256*256
COVID	URL	3616	6	0	https://sirm.org/category/senza-categoria/covi...
Lung_Opacity	FILE NAME	6012	6012	0	Lung_Opacity-1
Lung_Opacity	FORMAT	6012	1	0	PNG
Lung_Opacity	SIZE	6012	1	0	256*256
Lung_Opacity	URL	6012	1	0	https://www.kaggle.com/c/rsna-pneumonia-detect...
Normal	FILE NAME	10192	10192	0	NORMAL-1
Normal	FORMAT	10192	1	0	PNG
Normal	SIZE	10192	1	0	256*256
Normal	URL	10192	2	0	https://www.kaggle.com/c/rsna-pneumonia-detect...
Viral Pneumonia	FILE NAME	1345	1345	0	Viral Pneumonia-1
Viral Pneumonia	FORMAT	1345	1	0	PNG
Viral Pneumonia	SIZE	1345	1	0	256*256
Viral Pneumonia	URL	1345	1	0	https://www.kaggle.com/paultimothymooney/chest...

Dossier images et masks

data	dossier	N fichiers	N formats différents	Formats présents	exemple nom fichier
COVID	images	3616	1 [png]		COVID-1.png
COVID	masks	3616	1 [png]		COVID-1.png
Lung_Opacity	images	6012	1 [png]		Lung_Opacity-1.png
Lung_Opacity	masks	6012	1 [png]		Lung_Opacity-1.png
Normal	images	10192	1 [png]		Normal-1.png
Normal	masks	10192	1 [png]		Normal-1.png
Viral Pneumonia	images	1345	1 [png]		Viral Pneumonia-1.png
Viral Pneumonia	masks	1345	1 [png]		Viral Pneumonia-1.png

Comme les dossiers images et masks contiennent des fichiers .png avec le même nom, le fichier dans “masks” correspondant au fichier dans “images” du même nom, nous avons vérifié la cohérence entre les 2 dossiers pour chaque jeu de données.

	Nombre de fichiers dans images	Nombre de fichiers dans masks	Nombre d'images non retrouvés dans masks	Nombre de masks non retrouvés dans images
COVID	3616	3616	0	0
Lung_Opacity	6012	6012	0	0
Normal	10192	10192	0	0
Viral Pneumonia	1345	1345	0	0

	Nombre de fichiers dans images	Nombre de fichiers dans metadata	Nombre d'images non retrouvés dans metadata	Nombre de lignes de metadata non retrouvés dans images
COVID	3616	3616	0	0
Lung_Opacity	6012	6012	0	0
Normal	10192	10192	0	0
Viral Pneumonia	1345	1345	0	0

Les effectifs observés dans les données en fonction de la source référencée dans les metadata ont ensuite été comparés à ceux décrits dans le fichier README.md.

	Source README	Source metadata	Nb README	Nb observé
COVID	padchest dataset[1]	https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/#1590858128006-9e640421-6711	2473	2474
COVID	Germany medical school[2]	https://github.com/ml-workgroup/covid-19-image-repository/tree/master/png	183	183
COVID	SIRM, Github, Kaggle & Tweeter[3,4,5,6]	https://eurorad.org , https://sirm.org/category/senza-categoria/covid-19/ , https://github.com/ieee8023/covid-chestxray-dataset	559	560
COVID	another Github source[7]	https://github.com/armiro/COVID-CXNet	400	400
Normal	RSNA [8]	https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data	8851	8851
Normal	Kaggle [9]	https://www.kaggle.com/paultimothymooney/chest-x-ray-pneumonia	1341	1341
Lung opacity	Radiological Society of North America (RSNA) CXR dataset [8]	https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data	6012	6012
Viral Pneumonia	Chest X-Ray Images (pneumonia) database [9]	https://www.kaggle.com/paultimothymooney/chest-x-ray-pneumonia	1345	1345

[1]<https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/#1590858128006-9e640421-6711>

[2]<https://github.com/ml-workgroup/covid-19-image-repository/tree/master/png>

[3]<https://sirm.org/category/senza-categoria/covid-19/>

[4]<https://eurorad.org>

[5]<https://github.com/ieee8023/covid-chestxray-dataset>

[6]https://figshare.com/articles/COVID-19_Chest_X-Ray_Image_Repository/12580328

[7]<https://github.com/armiro/COVID-CXNet>

[8]<https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data>

[9] <https://www.kaggle.com/paultimothymooney/chest-x-ray-pneumonia>

2.2.2. COVID-QU-Ex Dataset

Pour le set “COVID-QU-Ex Dataset”, en l’absence de metadata accessible directement sur ce dataset, certaines caractéristiques ont été extraites à partir des fichiers .png, grâce aux librairies ‘opencv’ et ‘pillow’.

Lors de l’ouverture de chaque image, le nombre de canaux a été forcé à “1” pour tenir compte des images qui sont toutes en niveaux de gris (noir et blanc pour les masks). Ces caractéristiques comme les dimensions et la distribution de l’intensité peuvent apporter des premiers éléments de réflexion sur les traitements à faire lors de l’étape de pré-processing.

On remarque qu’il y a une dimension pour chaque type de .png : 299*299 pour les images et 256*256 pour les masks :

dossier	Hauteur	Largeur	N
images	299	299	21165
masks	256	256	21165

2.2.3. Conclusion sur l’intégrité

Pour le set “COVID-19_Radiography_Dataset”:

- 4 jeux de données : COVID, Lung_Opacity, Normal, Viral Pneumonia.
- Chaque jeu de données contient un fichier de metadata et 2 dossiers contenant des .png : "images" et "masks".
- Ensemble de données très propre : pas de Na, pas de doublons d'id, même structure pour les 4 jeux de données.
- Cohérence parfaite entre metadata, images et masks : chaque id est retrouvé dans les 3 sources.
- Globalement on retrouve les effectifs annoncés dans le README, à 1 ou 2 images près pour le jeu de données COVID.

Pour le set “COVID-QU-Ex Dataset”:

- Toutes les images ont le même format
- Tous les masques ont le même format, bien que différent de celui des images.

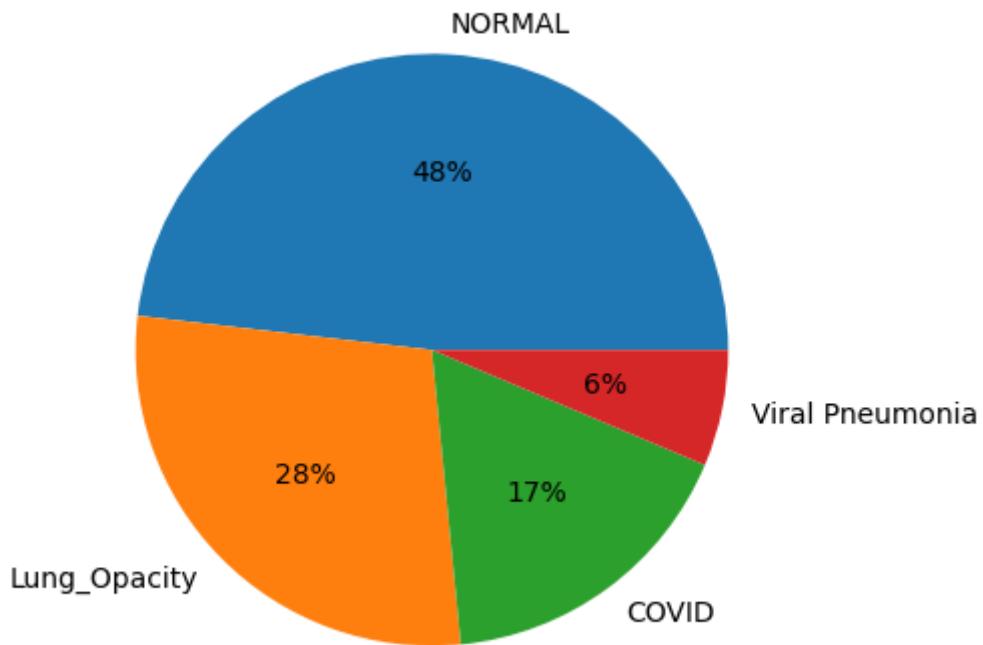
Pas d’autres informations extraites de ce jeu de données

2.3. Effectifs par type d'infection

2.3.1. COVID-19_Radiography_Dataset

Répartition des images

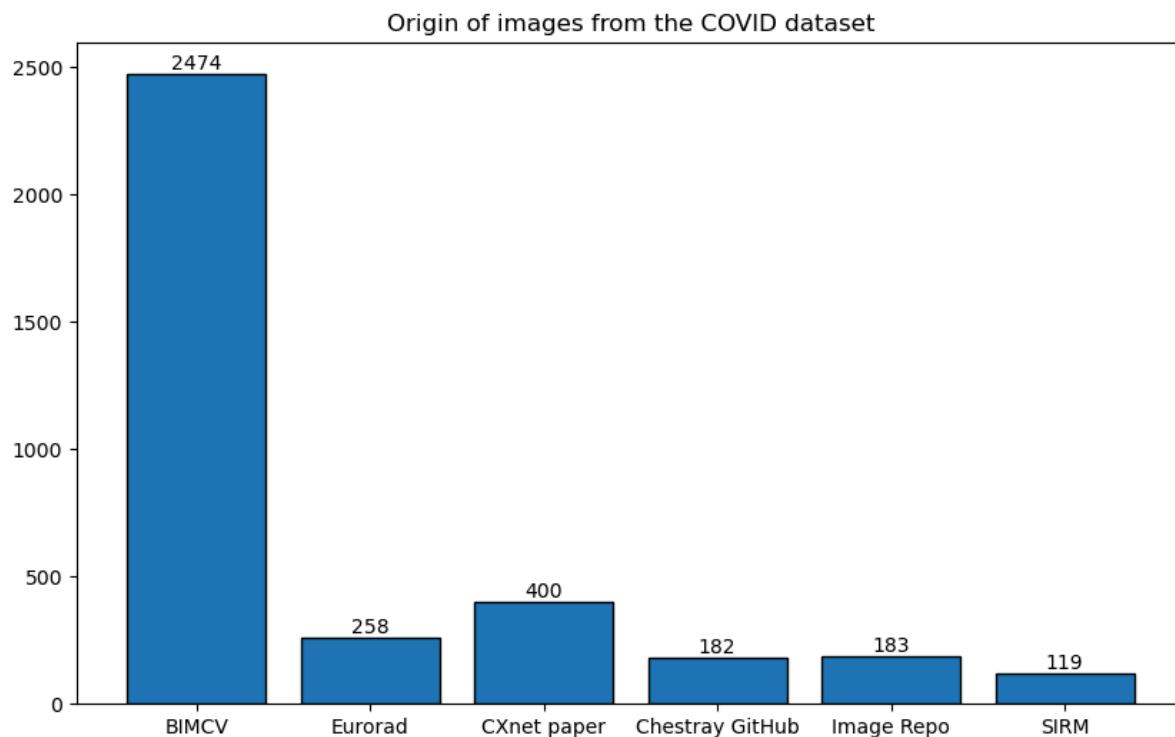
Répartition du type de radios



On observe ici une forte disparité entre les classes.

Presque la moitié des radios de ce jeu de données présentent des poumons sains (10192 images). En revanche il y a seulement 3616 radios de poumons labellisés “COVID”. Les autres infections, labellisées “Lung_Opacity” et “Viral Pneumonia”, sont au nombre de 6012 et 1345. Ainsi, **les cas de COVID représentent seulement 17% des cas de ce set.**

Origine des images du sous-dataset “COVID”



BMCV : Banco digital de Imagen Médica de la Comunidad Valenciana

SIRM : Società Italiana di Radiologia Medica e Interventistica

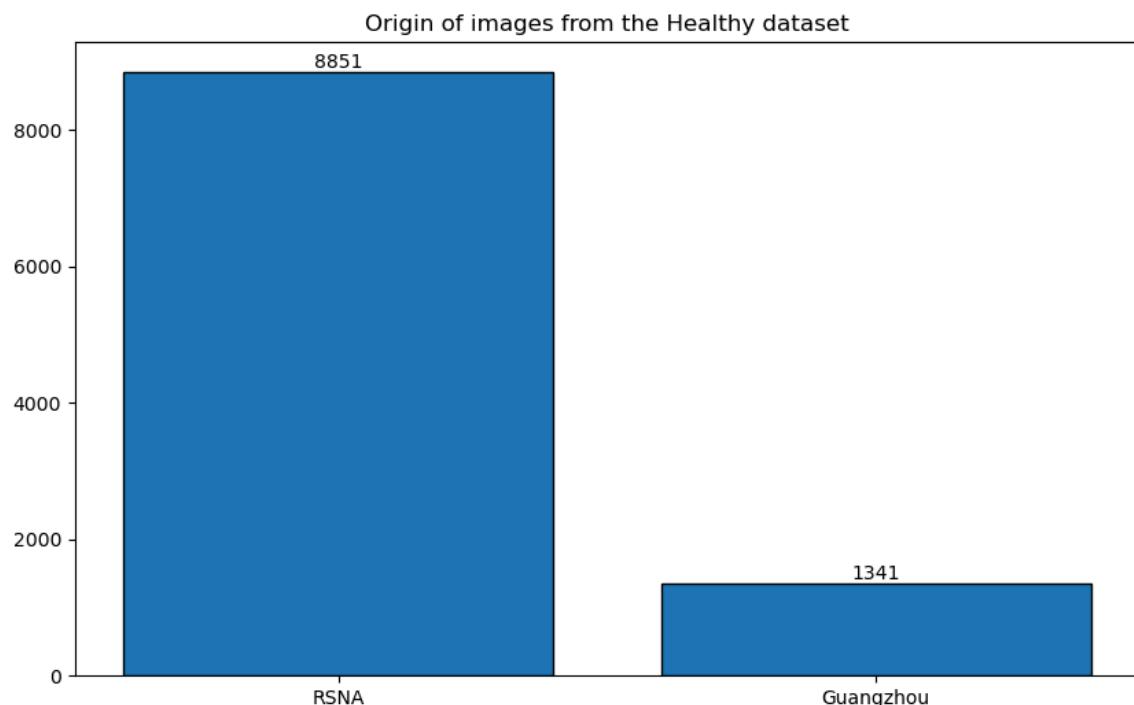
Eurorad : European Society of Radiology

Les trois autres sources des données (“CXNet paper”, “Chestray GitHub” et “Image Repo”) sont des regroupements de données **provenant de beaucoup de sources différentes**, y compris de BIMCV, SIRM et Eurorad.

La majorité des radios présentant des cas COVID proviennent d’Espagne. De plus, certaines sources étant elles mêmes des regroupement d’images, **il n'est pas exclu que certaines radios apparaissent plusieurs fois**. Il n'a pas été fait mention d'une attention particulière à ce détail dans la base de données.

Il est également possible qu'**un même patient ait fait plusieurs radios**. Dans ce cas, cela pourrait amener un biais dans un modèle, si celui-ci reconnaît plutôt un patient qu'un type d'infection. Cela peut également s'apparenter à une simple augmentation de données. Ainsi cela ne devrait pas poser de problèmes.

Origine des images du sous-dataset “Normal”

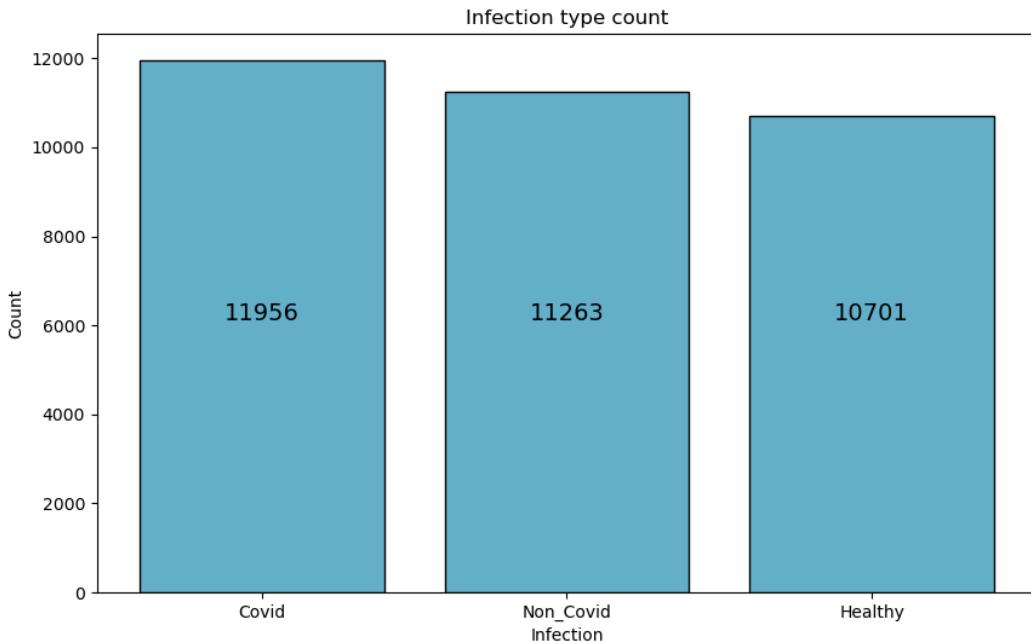


RSNA : Radiological Society of North America

Guangzhou: Guangzhou Women and Children's Medical Center, China

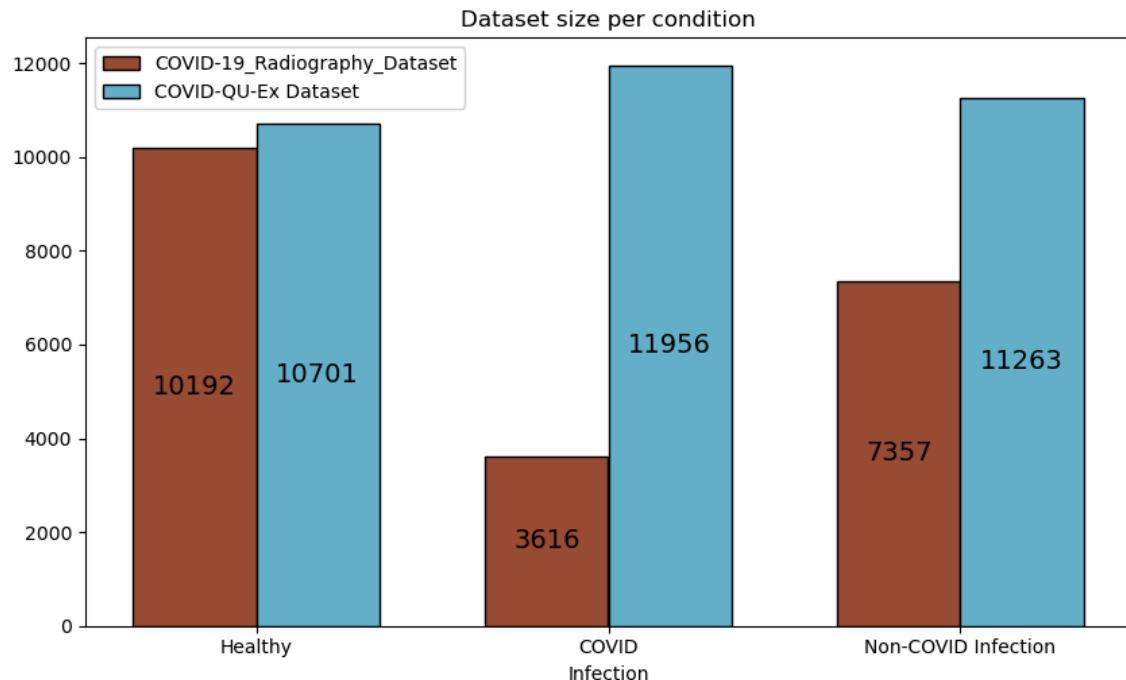
Les données provenant de Guangzhou Women and Children's Medical Center sont très certainement des **radios provenant d'enfants**, puisque c'est un centre médical de type maternité. Cependant, cela concerne moins de 15% des données du dataset de patients sains, **cela ne devrait donc pas introduire de biais** trop important dans un futur modèle.

2.3.2. COVID-QU-Ex Dataset



Dans ce dataset, il y a à peu près le même nombre d'images de personnes saines que dans le premier dataset. Cependant, il y a beaucoup plus de COVID (35%), ce qui est intéressant pour notre modélisation (données mieux équilibrées). Voici une comparaison entre les deux datasets :

(Lung_Opacity et Viral Pneumonia ont été réunies en Non-Covid Infections)



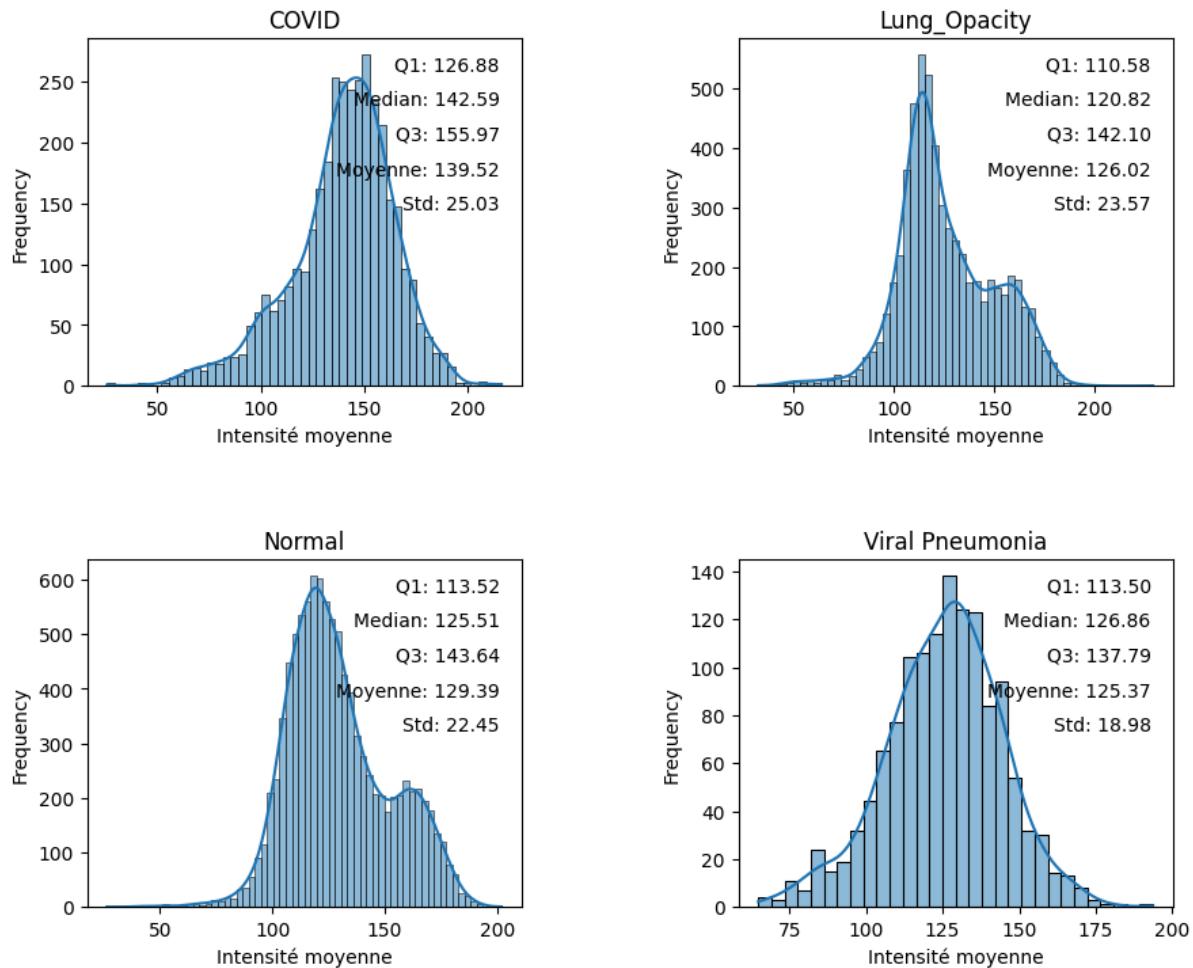
De part la proportion de données COVID, le nombre de données plus important ainsi que la date des dernières données (plus récentes), **il paraît plus intéressant d'utiliser le deuxième dataset, COVID-QU-Ex Dataset**

2.4. Intensités de pixels moyennes

2.4.1. COVID-19_Radiography_Dataset

Images

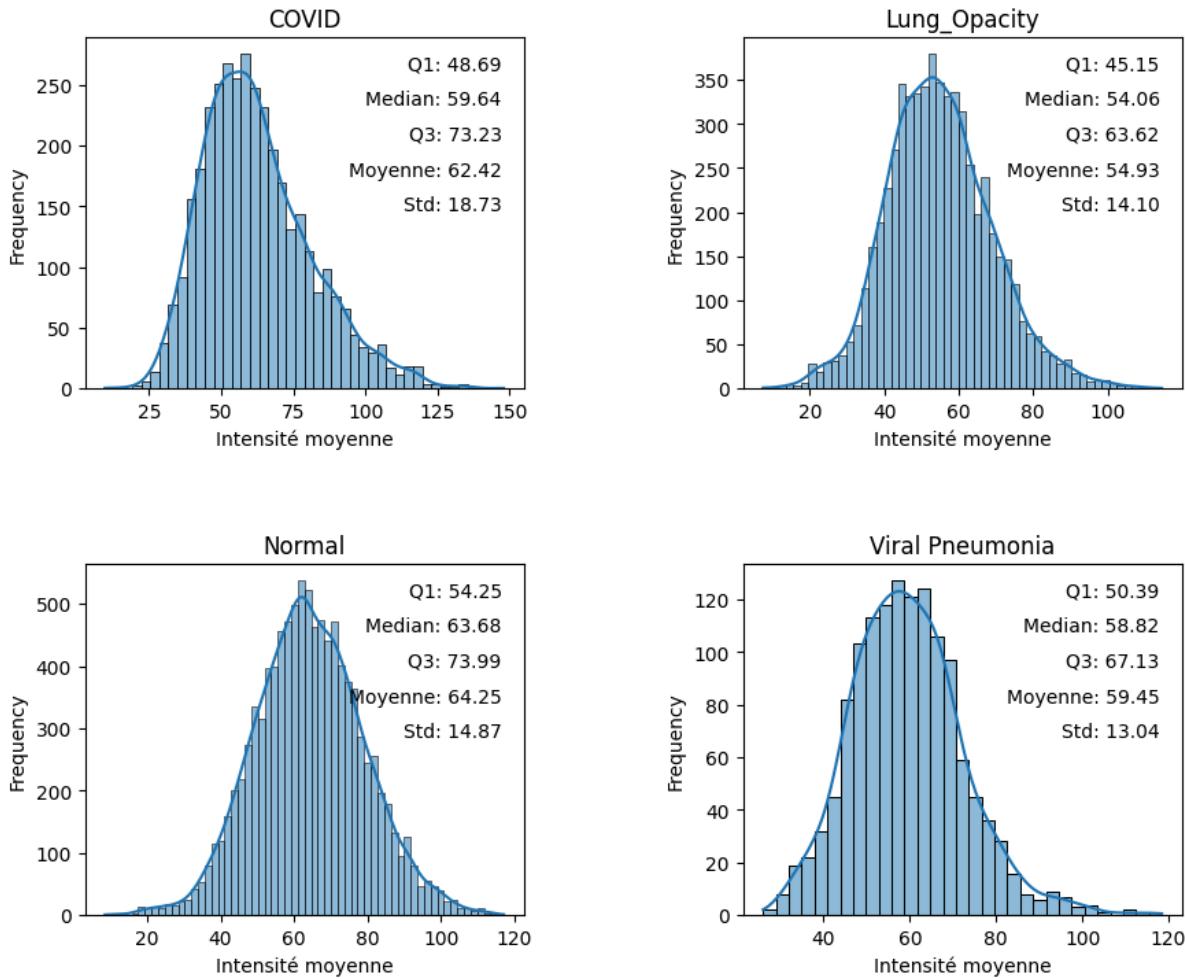
Distribution de l'intensité de pixels moyenne pour les fichiers 'images'



On observe une **moyenne d'intensité plus importante** pour les images labellisées “**COVID**” que pour les 3 autres labels. Les infections faisant apparaître les poumons beaucoup plus clairs sur les radios, il semble logique que l'intensité moyenne soit plus élevée. Cependant cela devrait seulement donner une intensité plus faible pour les poumons sains, et plus élevée pour les autres. On voit donc ici que les cas de COVID se démarquent du reste des infections.

Masks

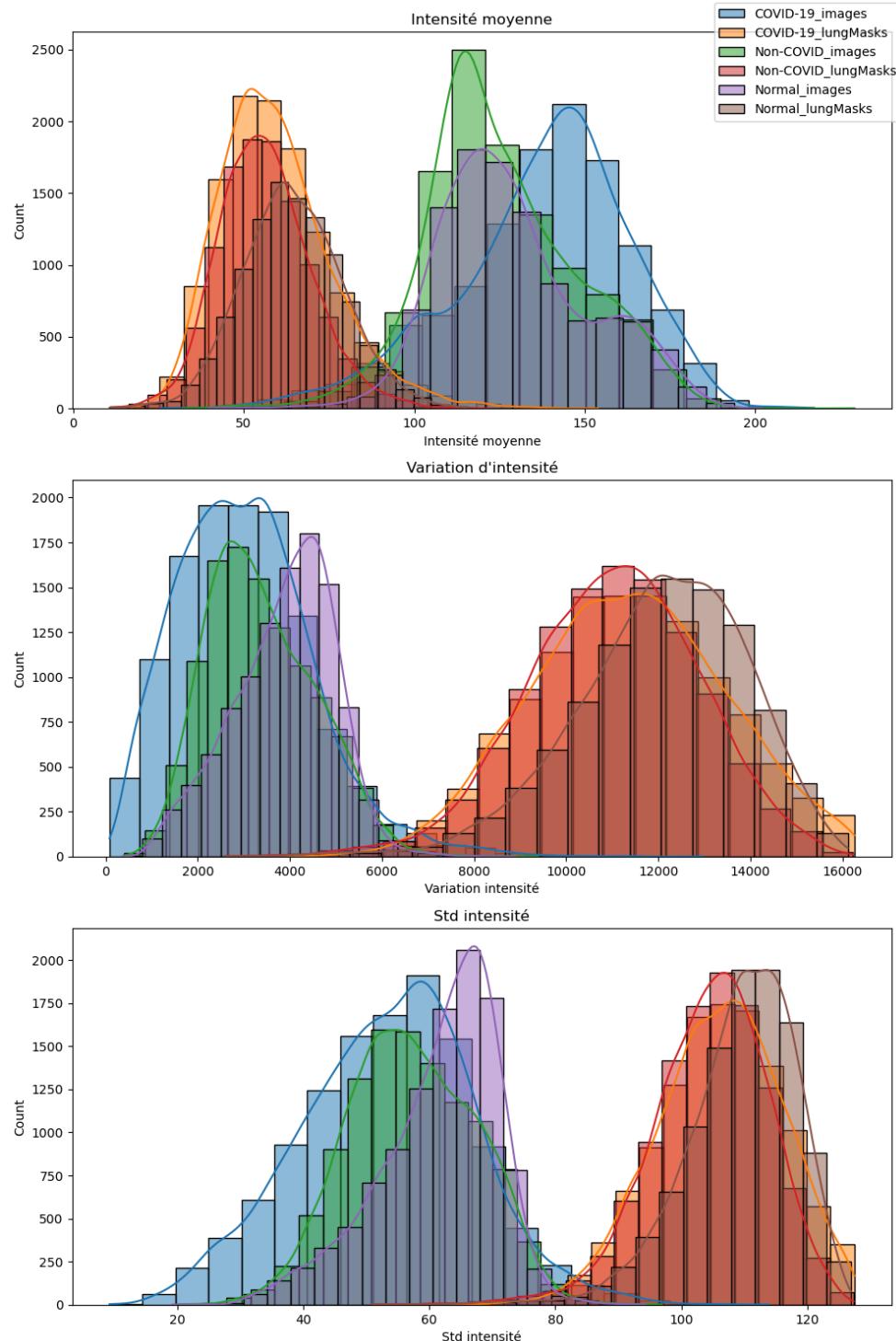
Distribution de l'intensité de pixels moyenne pour les fichiers 'masks'



La différence est moins marquée ici, ce qui paraît logique, l'intensité est liée en grande partie à la taille des poumons, qui ne devrait pas être liée au type d'infection. Évidemment, **l'intensité est bien moins importante** ici puisque pour tout ce qui ne concerne pas les poumons (c'est à dire la majorité de l'image), l'intensité a été réduite à zéro. Nous avons donc des intensités moyennes deux fois plus faibles que l'intensité des images radios.

2.4.2. COVID-QU-Ex Dataset

Répartition de l'intensité moyenne, de la var et de la std sur le jeu de données QU-ex, pour “lung segmentation” :



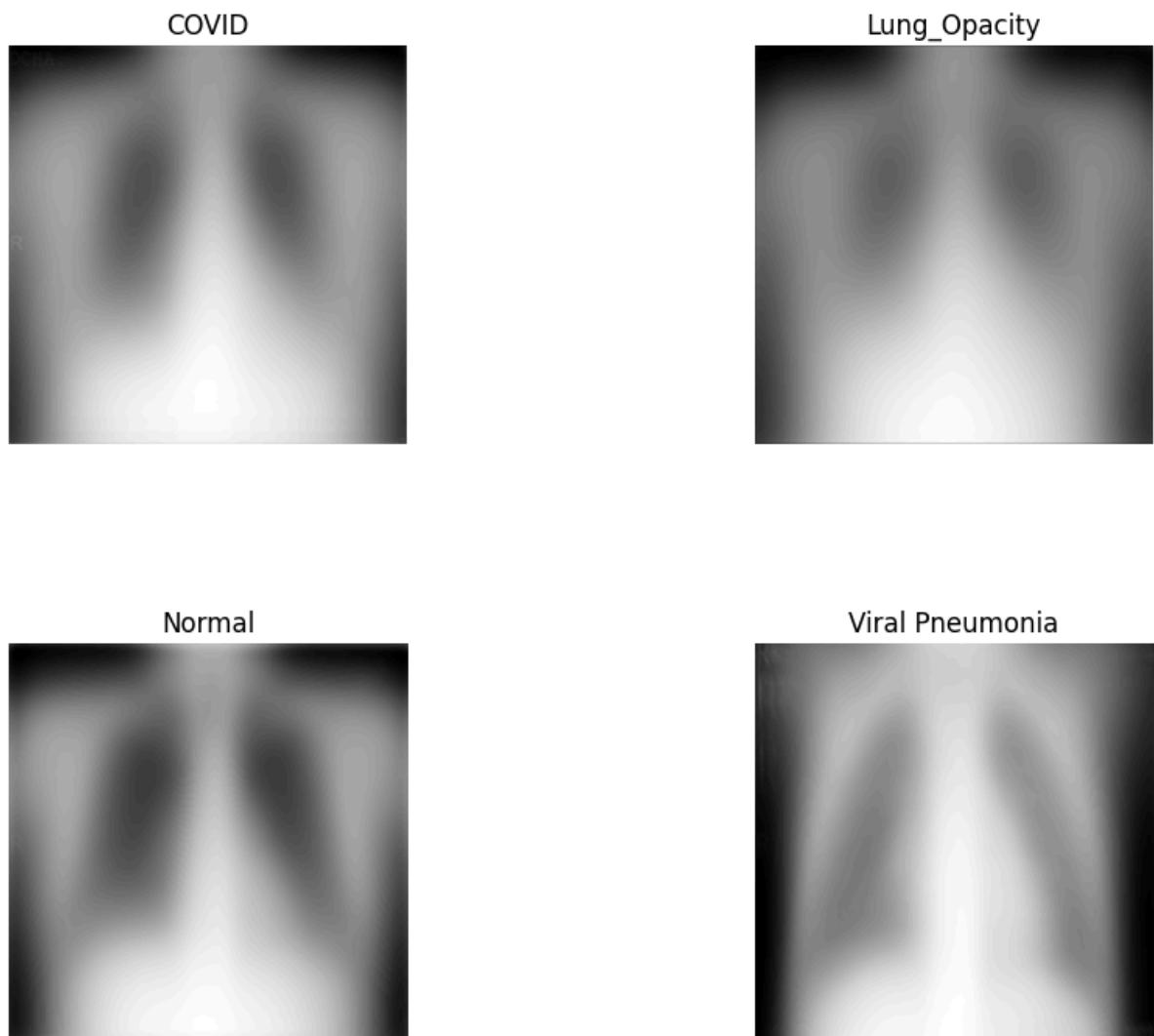
Le constat est similaire ici, les images “COVID” sont plus intenses que les autres. En revanche, l’écart type est similaire entre les différents types de données.

2.5. Images moyennes

Des images moyennes ont été calculées pour chaque jeu de données et chaque type de png du jeu de données “COVID-19_Radiography_Dataset”. Certaines radios n'étant pas centrées, d'autres rétrécies ou tournées, l'image moyenne n'est pas extrêmement pertinente.

Images

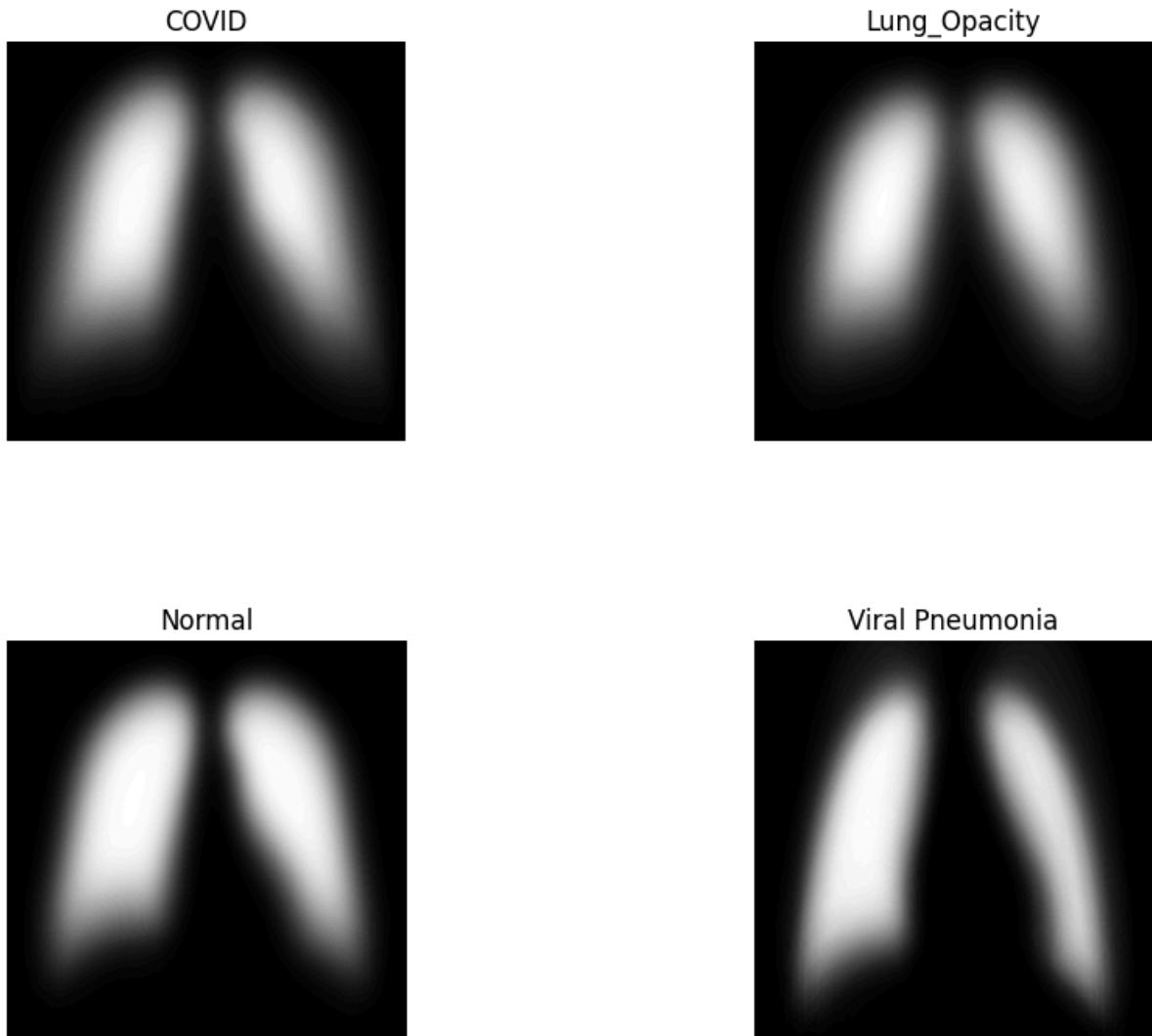
Images moyenne pour les fichiers 'images'



Les formes de poumons semblent un peu différentes, mais il est difficile de dire à ce niveau si c'est une tendance, ou un biais, dû au nombre d'images ou à la source de celles-ci par exemple..

Masks

Images moyenne pour les fichiers 'masks'



Les images moyennes des radios et des masques sont cohérentes entre elles.

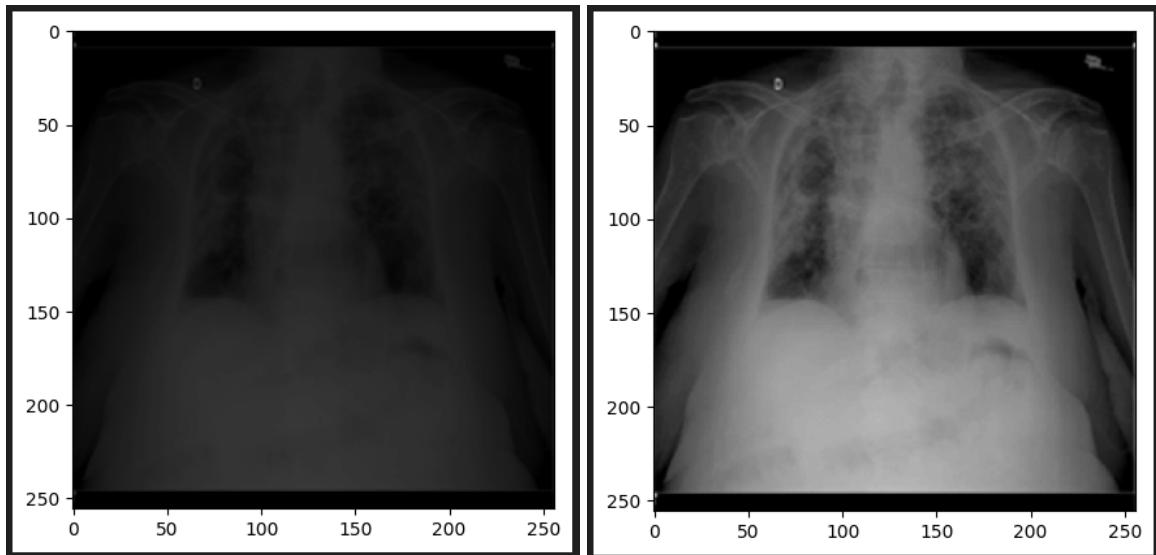
2.6. Conclusion sur le choix du jeu de données

L'analyse des données et leur visualisation nous mène à choisir le set “**COVID-QU-Ex Dataset**”, qui n'est donc pas le set initial, comme base pour construire notre modèle. Bien que manquant de certaines sources, l'équilibre entre les différents labels, ainsi que le nombre d'images plus important nous amène à penser que le modèle en résultant sera plus robuste. De plus, cela n'a pas été fait par les précédents groupes, cela amènera donc une autre vision de la problématique.

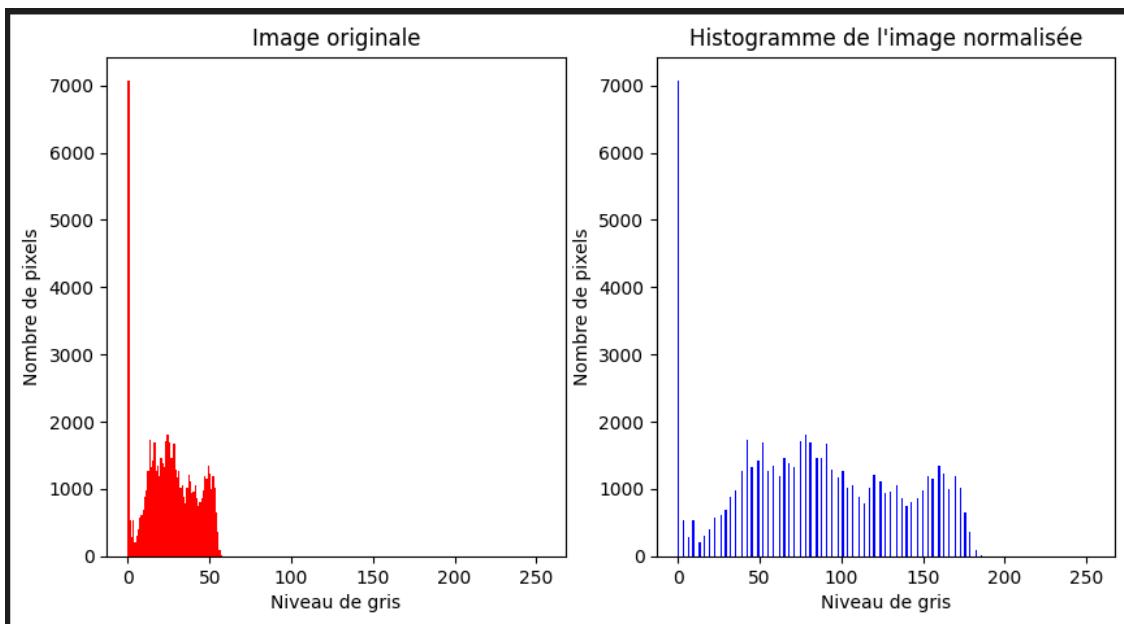
3. Preprocessing des images

3.1. Normalisation Min-Max

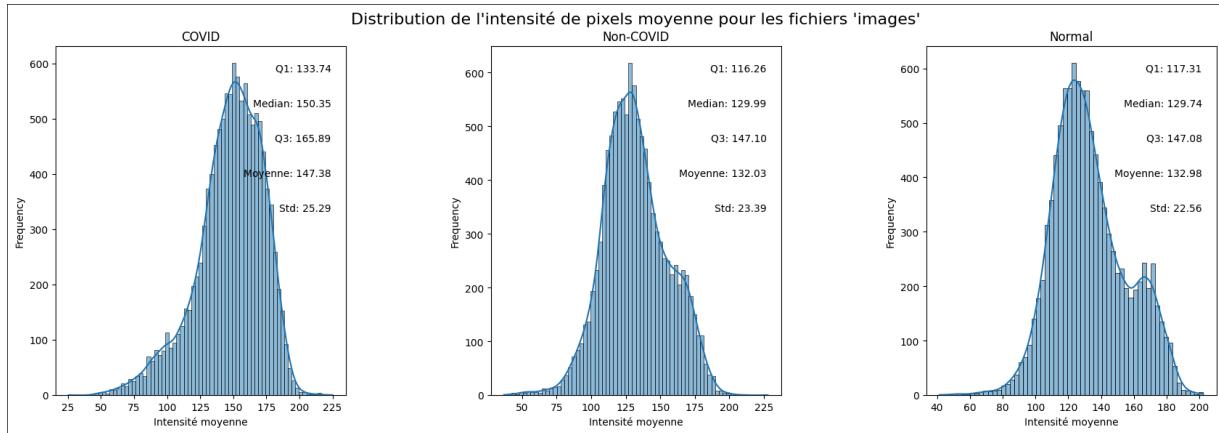
La première normalisation envisagée est la normalisation MinMax [15] . On y observe principalement un **éclaircissement des images**. Voici un exemple sur l'image Covid-200, qui est un cas extrême:



L'image résultante est en effet beaucoup plus claire, cependant en regardant la distribution de l'intensité des pixels sur l'image, on se rend compte que nous n'utilisons toujours pas vraiment la totalité de la plage d'intensité. Dans ce cas précis, il doit y avoir un pixel qui était beaucoup plus clair que les autres, et la normalisation résultante n'est pas optimale.



Distribution des intensités après normalisation:

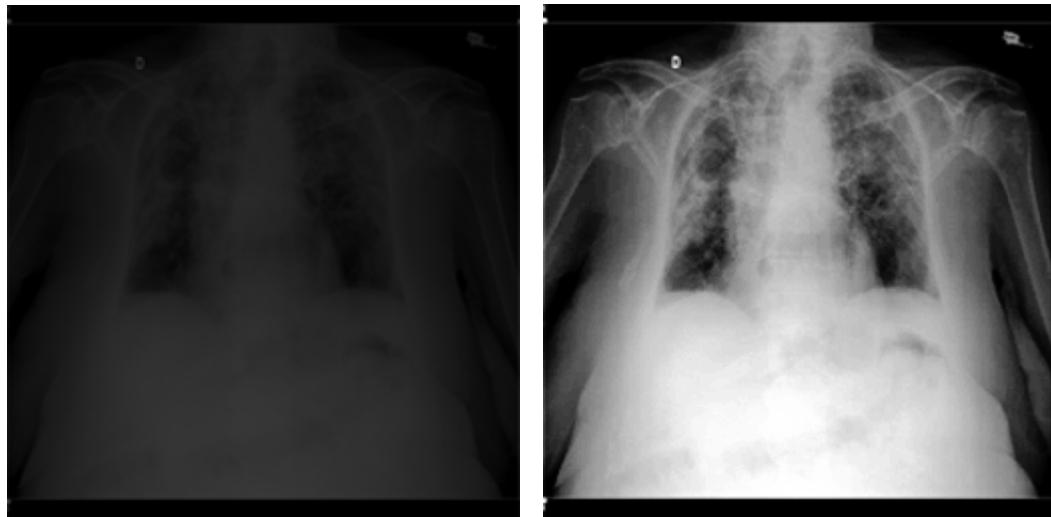


On remarque ainsi des **intensités moyennes plus importantes** que précédemment. De plus, l'intensité moyenne pour le label COVID a augmenté plus que les autres, ce qui est intéressant car cela pourrait aider le modèle à détecter les cas positifs au COVID.

(Les intensités moyennes avant normalisation étaient de 137.6 pour le label “COVID”, 125.6 pour le label “Non-COVID”, et 129.27 pour le label “Normal”)

3.2. Normalisation par égalisation d'histogrammes

La seconde méthode envisagée est la normalisation par égalisation d'histogrammes. En effet, après recherche bibliographique, cela semble être une **méthode très utilisée** en imagerie médicale [16]. On y observe un **fort contraste** par rapport à la normalisation Min-Max. Voici un exemple sur la même image que précédemment:



Le contraste est en effet meilleur, cela semble être aussi une bonne méthode de normalisation. Le problème soulevé rencontré en Normalisation Min-Max semble avoir disparu ici.

3.3. Application des masques sur les images

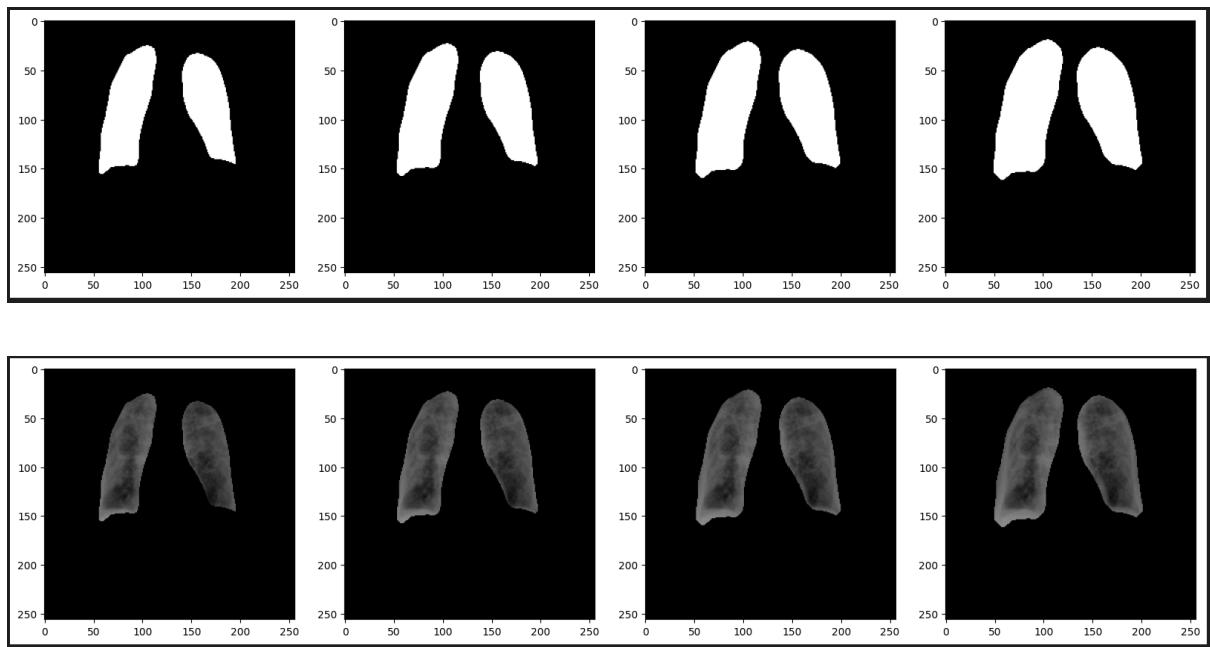
Lors du preprocessing, nous avons également décidé de générer les images après application des masques, pour pouvoir tester ces données là pendant avec nos modèles de deep-learning. Après un premier test, nous avons trouvé les **masques un peu restrictifs**, donc nous avons également décidé d'appliquer une dilatation dessus, afin d'étudier les performances. Plusieurs dilatations différentes ont été essayées.

3.3.1. Dilatations successives.

Nous avons commencé par une dilatation avec un noyau de taille moyenne appliqué 1, 2 ou 3 fois. Le noyau est le suivant :

```
[0, 0, 1, 0, 0]  
[0, 1, 1, 1, 0]  
[1, 1, 1, 1, 1]  
[0, 1, 1, 1, 0]  
[0, 0, 1, 0, 0]
```

On voit ainsi la modification apportée à un masque après les différentes itérations, puis le résultat après application sur l'image:

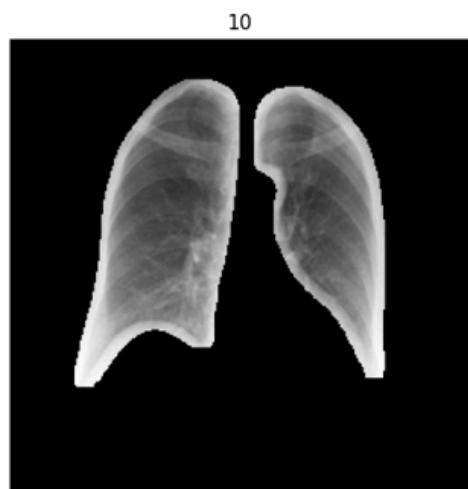
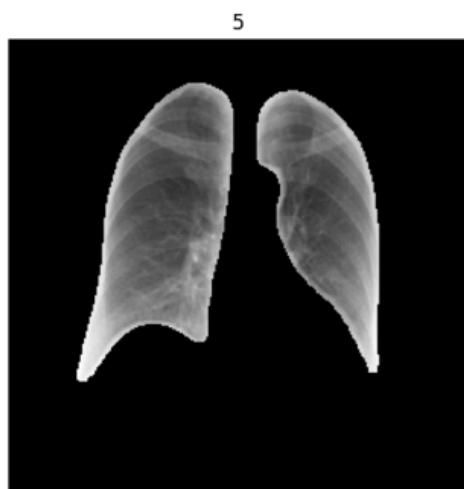
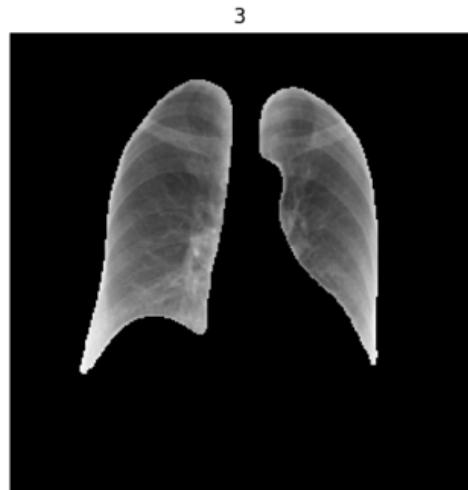
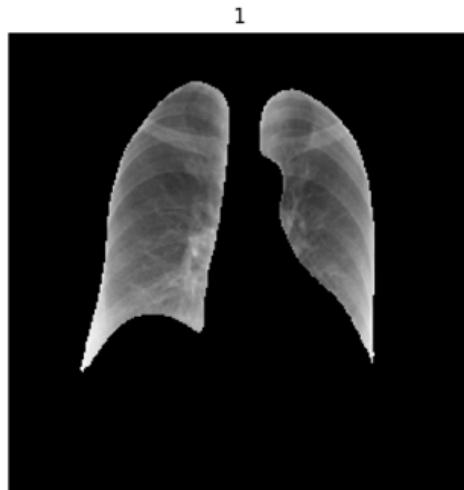


Dès la première dilatation avec ce noyau, on peut voir la majorité des contours des poumons, cela semble être suffisant pour ne pas perdre d'information. Les masques semblent donc être faits au pixel près.

3.3.2. Dilatation avec noyaux de différentes tailles

Nous avons également réalisé des dilatations avec des noyaux très simples, carrés, de côté 1, 3, 5 et 10 pixels. Les images suivantes ont été réalisées sur des images normalisées avant application des masques.

Test kernels carrés pour COVID / sub-S09955_ses-E17098_run-1_bp-chest_vp-pa_dx.png



On voit ici aussi qu'un noyau de taille 3 ou 5 est suffisant pour voir les contours des poumons. En revanche, le noyau de taille 10 est beaucoup trop grand

4. Première approche du deep-learning

4.1. Fonctionnement d'un modèle de deep learning

Un modèle de deep learning est une architecture de réseau de neurones artificiels conçue pour apprendre et effectuer des tâches complexes en analysant des données massives. À la base, un modèle de deep learning se compose de couches de neurones artificiels, organisées en une structure appelée réseau de neurones profond. Chaque neurone dans une couche est connecté à chaque neurone de la couche suivante, formant ainsi un **réseau dense**.

Parmi les architectures couramment utilisées, les **réseaux de neurones convolutifs (CNN)** sont particulièrement efficaces pour le traitement des données visuelles. Les CNN utilisent des couches de convolution pour extraire des caractéristiques pertinentes des images, réduisant ainsi le besoin de prétraitement manuel.

Le processus d'apprentissage d'un modèle de deep learning commence par l'initialisation des poids des connexions entre neurones. Ensuite, les données d'entraînement sont introduites dans le réseau, couche par couche. À chaque couche, les neurones appliquent des transformations mathématiques aux données, notamment des multiplications de matrices et des ajouts de biais, suivies par une fonction d'activation (comme ReLU ou sigmoid) qui introduit de la non-linéarité. Les résultats passent d'une couche à l'autre jusqu'à la couche de sortie, où le modèle produit une prédiction.

Pour évaluer la performance d'un modèle de deep learning, on utilise diverses métriques en fonction de la tâche de classification. Dans les tâches de **classification binaire**, où il s'agit de distinguer entre deux classes (par exemple, spam ou non spam), des métriques comme l'**accuracy** (que l'on conservera en anglais pour ne pas confondre avec la "precision" vue ensuite), la **precision**, le **recall** (rappel) et le **f1-score** sont couramment utilisées. L'accuracy mesure la proportion de prédictions correctes sur l'ensemble des prédictions. La précision indique la proportion de prédictions positives correctes par rapport au nombre total de prédictions positives. Le rappel, quant à lui, mesure la proportion de prédictions positives correctes par rapport au nombre total de véritables instances positives. Le score F1 est la moyenne harmonique de la précision et du rappel, fournissant une mesure équilibrée entre les deux.

Pour les **tâches de classification multiple**, où le modèle doit distinguer entre plus de deux classes (par exemple, reconnaissance de chiffres manuscrits de 0 à 9), on utilise également l'accuracy ainsi que des métriques comme la **moyenne de précision pondérée** et la **moyenne de rappel pondérée**, qui prennent en compte l'importance relative de chaque classe.

Pour améliorer la performance du modèle, le processus d'apprentissage inclut une étape de rétropropagation, où l'erreur de prédiction est calculée à l'aide d'une fonction de coût (comme l'entropie croisée pour la classification) et les gradients sont propagés en arrière à travers le réseau pour ajuster les poids des connexions. Ce processus est itératif et se répète jusqu'à ce que le modèle converge vers une solution optimale.

4.2. Application à notre jeu de données

Dans notre cas, nous devons classifier nos données selon 3 labels. Nous avons donc une **classification multiple** à réaliser. L'évaluation de nos modèles a été faite principalement selon les valeurs de l'**accuracy** (proportion de prédictions correctes parmi toutes les prédictions), dans le but de prédire avec certitude l'état des patients. Le fait d'avoir des classes équilibrées a également conforté notre choix.

Nous avons ensuite généré des rapports de classification dans lesquels nous avons étudié :

- la **précision** qui mesure la proportion de véritables positifs parmi les cas prédis comme positifs, pour s'assurer que les faux positifs sont minimisés
- le **recall** qui mesure la proportion de véritables positifs correctement identifiés parmi tous les cas réels de la classe considérée, pour s'assurer que les patients positifs (par exemple au COVID) ne sont pas manqués.
- le **f1-score** qui combine la précision et le recall en une seule métrique, et donne ainsi une mesure équilibrée de la performance du modèle.

Enfin pour avoir une vue d'ensemble détaillée des performances de notre modèle, nous avons généré la **matrice de confusion**, qui montre les vrais positifs, vrais négatifs, faux positifs et faux négatifs pour chaque classe.

Dans un premier temps, comme il est courant de le faire en deep learning, nous avons utilisé un modèle LeNet pour un premier test de prédictions. Cela nous permet également de comparer les différents preprocessing appliqués à nos images pour décider de ce que nous utiliserons par la suite.

4.3. Modèle LeNet

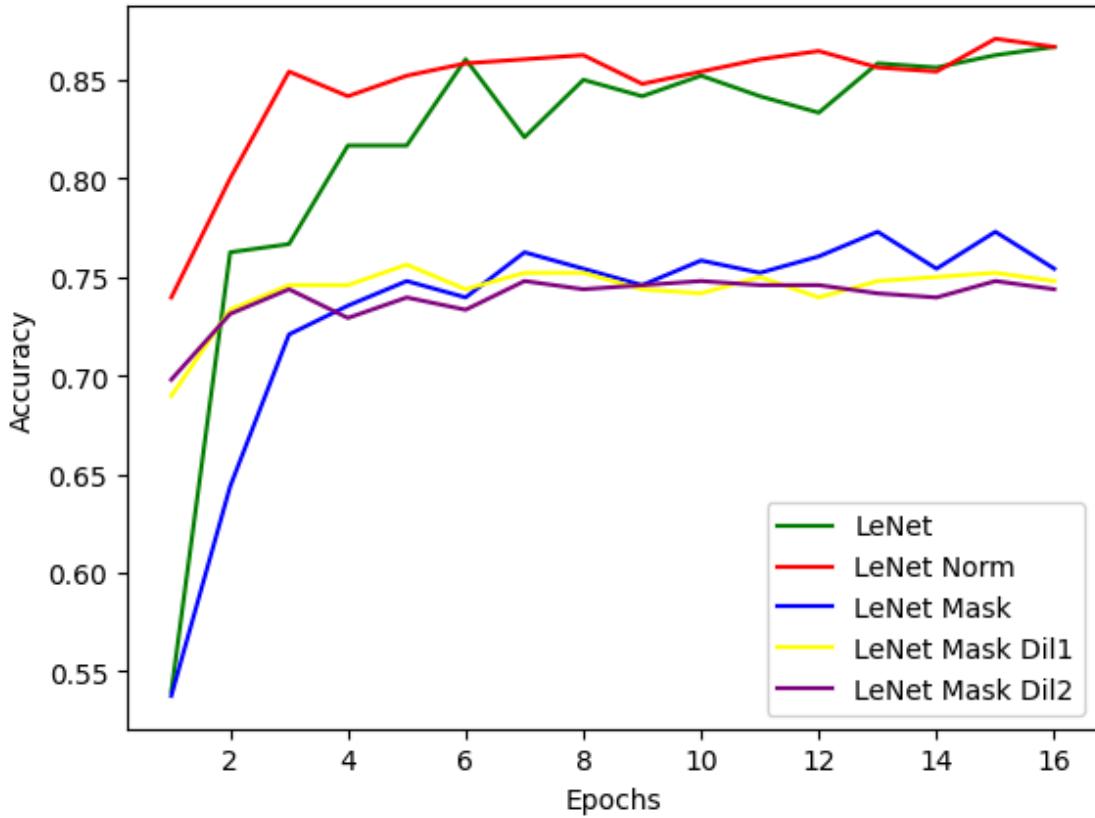
4.3.1. Description du modèle

Le modèle LeNet-5 est un modèle CNN simple créé en 1998. Il est constitué de 2 sets de couches de convolutions suivis après aplatissement, de 2 couches denses avant classification finale. C'est un modèle reconnu pour son efficacité très correcte avec des temps d'entraînements assez courts, de part son nombre de couches, seulement 7.

4.3.2. Déroulement de l'entraînement

Le modèle a d'abord été entraîné sur les images brutes. On obtient ainsi un **accuracy de 85%**. Une comparaison a été réalisée avec les images normalisées via une normalisation Min-Max, avec une accuracy de 86%.

Afin d'évaluer l'impact des masques sur les radios, le modèle a également été testé sur les images brutes avec le masque appliqué tel quel, ou dilaté une ou plusieurs fois.

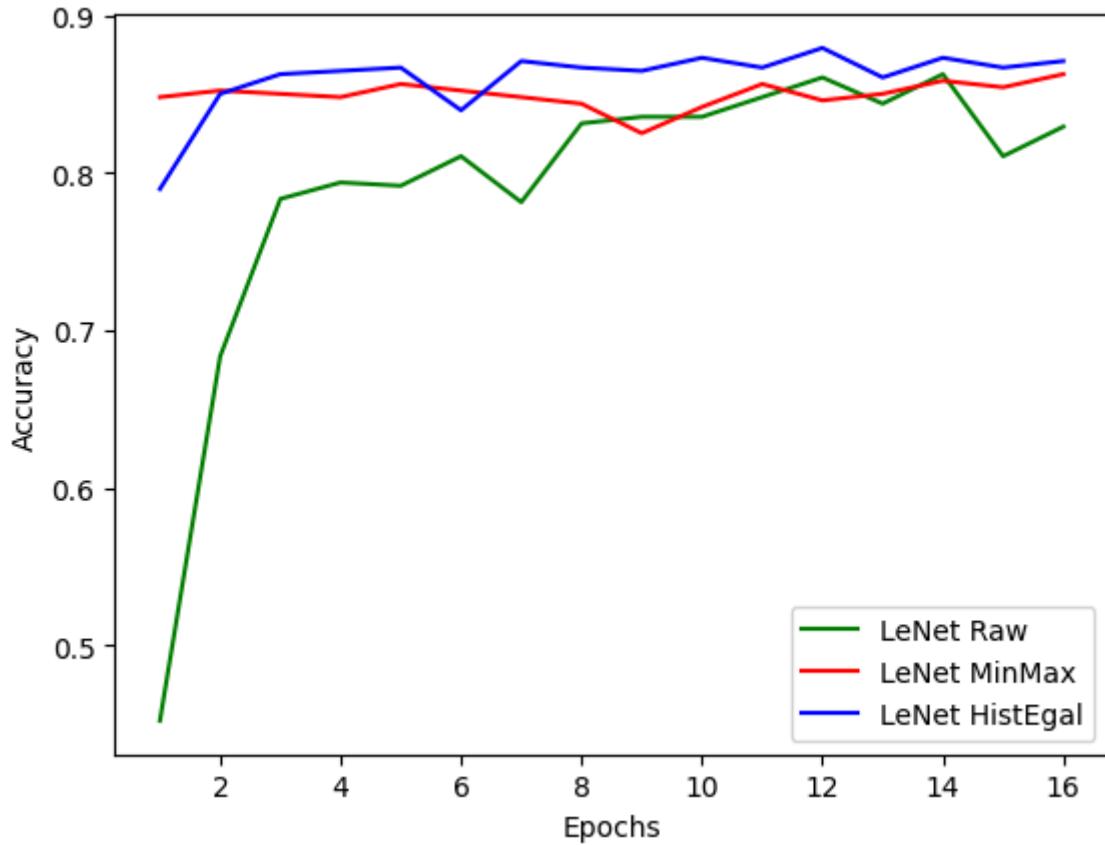


On obtient ainsi une accuracy de 72% sur les masques appliqués tels quels, 72% lorsqu'ils sont dilatés une fois, et 74% lorsqu'ils sont dilatés deux fois. Plus les masques sont dilatés, plus on se rapprochera de l'image de base, il est donc normal que l'accuracy augmente. Cependant, les masques perdent leur intérêt s'ils sont beaucoup dilatés et qu'on obtient beaucoup d'informations hors des poumons.

Bien qu'ayant une accuracy très faible en première intention, nous avons décidé de tester notre jeu de données avec les images masquées également en transfer learning. En effet, une rapide inspection des radios montre de nombreuses annotations, flèches... Cela a de fortes chances de perturber notre modèle qui ne cherchera pas des signes d'infection, mais des éléments introduits a posteriori sur la radio. Afin de conserver **l'intérêt scientifique** de notre modèle, **les masques semblent donc essentiels**. Nous conserverons les masques originaux, puisque cela ne semble pas affecter l'accuracy outre mesure.

Après une étude plus approfondie des différentes techniques de normalisation, le modèle a été entraîné à nouveau en prenant en compte la normalisation par égalisation des histogrammes. Cette normalisation est celle qui visuellement donne les radios les plus lisibles avec un bon contraste.

On peut observer une comparaison entre les images brutes et les deux techniques de normalisation sur le graphe suivant:



Les deux types de normalisation menant aux mêmes résultats, c'est la normalisation par égalisation des histogrammes qui a servi de base pour l'entraînement des modèles de transfer learning, pour la clarté des images obtenues. Par la suite, nous avons rencontré la **normalisation CLAHE** [12], [17] , qui est une variante de la normalisation par égalisation d'histogramme (Contrast Limited Adaptive Histogram Equalization). Plusieurs modèles sont donc également normalisés par CLAHE dans la partie de Transfer Learning.

5. Transfer Learning

Après un premier test de modèle concluant via un LeNet-5, on passe à l'étape de Transfer Learning. Il s'agit de sélectionner plusieurs **modèles pré-entraînés** sur la base de données ImageNet afin d'obtenir la meilleure accuracy possible. Cela permet d'obtenir des résultats solides en conservant des **temps d'entraînement relativement courts**. La problématique de temps est en effet cruciale lors de l'entraînement de modèles de deep learning.

Afin de sélectionner les modèles les plus adaptés, nous avons recherché quels modèles ont déjà été utilisés et décrits dans le cadre d'imagerie médicale. Plusieurs papiers décrivent l'utilisation du transfer learning dans le milieu médical, mais principalement à partir de CT scans (Computed Tomography) [18], [19]. Bien que différents des scans X-Ray que nous possédons, cela reste bien plus proche de nos données que la base ImageNet.

Nous avons ainsi plusieurs modèles qui ressortent en imagerie :

- Xception
- MobileNet-V2
- EfficientNet (Plusieurs modèles)
- DenseNet (Plusieurs modèles)
- VGG (Plusieurs modèles)
- ResNet

Un modèle nommé ChexNet a également été utilisé pour des données X-Ray de poumons [20]. Malheureusement, c'est un modèle qui n'est pas disponible nativement dans nos librairies, nous avons donc décidé de ne pas nous pencher dessus. Nous avons en effet déjà suffisamment de modèles à tester en première intention.

5.1. Modèle Xception

5.1.1. Description du modèle

Le modèle Xception a été créé par le créateur de la bibliothèque Keras en 2017 [21]. C'est une variation plus performante des modèles Inception classiques, sans augmenter les temps de calcul.

C'est un modèle composé de 132 couches:

- L'entrée "Entry Flow", constituée de 35 couches
- Le "Middle Flow" composé de 80 couches (10 couches répétées 8 fois)
- La sortie "Exit Flow", composée de 17 couches.

5.1.2. Déroulement de l'optimisation

Keras Tuner et dégel des couches du modèle de base

Dans un premier temps, le modèle a été utilisé pré-entraîné avec les données imagenet. Les couches du modèle Xception ont été gelées pour ne pas les réentraîner.

Les couches finales de classification appliquées sont standard:

- GlobalAveragePooling2D
- Couche dense 1024 neurones avec activation 'relu'
- Dropout de 20%
- Couche dense de 512 neurones avec activation 'relu'
- Dropout de 20%
- Dernière couche Dense de Classification avec 3 neurones pour les 3 classes du modèle, avec une activation 'softmax'

On obtient ainsi **une accuracy de 89.1%** sur 10 époques sur les images normalisées par égalisation des histogrammes.

Pour commencer l'optimisation, un keras tuner sur 5 époques a été réalisé pour déterminer les taux de DropOut les plus performants. Les couches denses contenaient respectivement 256 et 128 neurones.

Les résultats ont montré que chaque combinaison de DropOut rate n'a pas influé de manière décisive sur l'accuracy, qui est restée entre 85 et 86%. Pour pouvoir comparer objectivement ce résultat avec le précédent, il aurait été pertinent de réaliser le keras tuner sur 10 époques avec le même nombre de neurones sur les couches Dense. Cela n'a pas été fait pour une question de temps, mais cela diminue grandement l'intérêt du keras tuner rétrospectivement.

Le premier test de dégel des couches du modèle de base s'est révélé probant. En dégelant l'ensemble du "Middle Flow" et de l'"Exit Flow", on atteint ainsi **une accuracy de 94.1%**. A noter que l'entraînement du modèle est assez court, puisqu'une époque prend en moyenne moins de 2 minutes à tourner.

Approfondissement du dégel et comparaison avec les images masquées

Pour avoir des résultats fiables à comparer, il a été décidé de faire systématiquement tourner les modèles sur 20 époques à partir de cette étape.

En ce qui concerne les neurones des couches dense, un compromis a été trouvé : 512 neurones pour la première couche Dense de classification, 256 pour la deuxième.

En effet, sur 20 époques avec 1024/512 neurones, on commence à observer de l'overfitting sur le modèle. Cependant, 256/128 neurones s'avéraient trop peu pour les 30 000 images en entrée, avec donc une performance moins optimale.

Avec les paramètre décrits ci-dessus, le modèle avec dégel des couches intermédiaires et finales atteint une **accuracy de 95.2%**.

Afin d'éventuellement gagner du temps sur l'entraînement, un test a été réalisé en ne dégelant que la boucle de sortie ("Exit Flow") du modèle. On obtient ainsi une accuracy de 93.8%. Le gain de temps est visible (environ 20% plus rapide à entraîner), mais la performance a été privilégiée. En effet, ce modèle possède déjà un temps d'entraînement relativement court.

A partir de ce modèle qui dépasse les 95% d'accuracy, une comparaison a été faite avec les **masques appliqués** sur les images pour ne faire ressortir que les poumons. Comme vu précédemment avec le modèle LeNet, on s'attend à une chute de performance. On obtient ainsi une **accuracy de 89.1%**, ce qui reste très correcte pour des images en entrée contenant beaucoup moins d'informations. Cela est certainement dû à la base de données utilisée qui est beaucoup plus grande et équilibrée que la base de données initiale. Cependant, on observe un léger overfitting avec ces données. En effet, les images contenant beaucoup moins d'informations, il devient plus facile pour le modèle d'être surentraîné.

Insertion de callbacks

Afin d'améliorer les performances du modèle avec les images masquées, pour lequel on observe de l'overfitting, le taux de dropout a été augmenté à 50% sur les deux dernières couches. Malheureusement, cela abaisse la performance à 87.2%, sans éliminer l'overfitting.

En ajoutant deux callbacks, EarlyStop et ReduceLROnPlateau, on remonte à 89.5% d'accuracy, mais l'overfitting est toujours présent sur les images masquées.

Après plusieurs tests sur les images non masquées, l'EarlyStop a été retiré. En effet, l'entraînement s'arrête parfois tôt à cause des fluctuations de la perte des données de validation (`val_loss`), menant à de mauvaises performances. De plus puisque l'overfitting reste présent sur les images masquées, il a été jugé non nécessaire.

L'entraînement du modèle avec un seul callback sur les images non masquées donne ainsi une **accuracy de 96.1%**. C'est ce modèle qui est retenu pour réaliser les GradCam, ainsi que pour comparer avec les autres modèles de transfer learning.

5.1.3. Interprétabilité des modèles - GradCam

Afin d'utiliser les algorithmes de GradCam proposés par Keras, les modèles ont été recréés de manière fonctionnelle et non séquentielle. L'accuracy des modèles réentraînés est cohérente avec ce qui a été décrit précédemment (96.0% pour les images non masquées, 90.3% pour les images masquées). On obtient ainsi les GradCam présentées ci-dessous.

Image 1 - Predicted label: Non COVID | True label: Non COVID

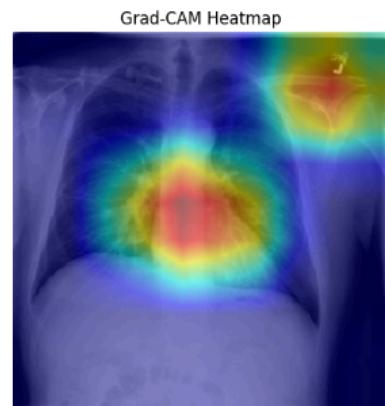
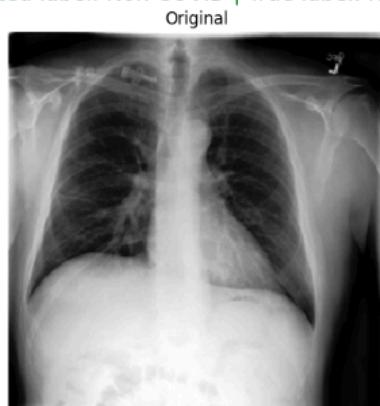


Image 5 - Predicted label: COVID | True label: COVID

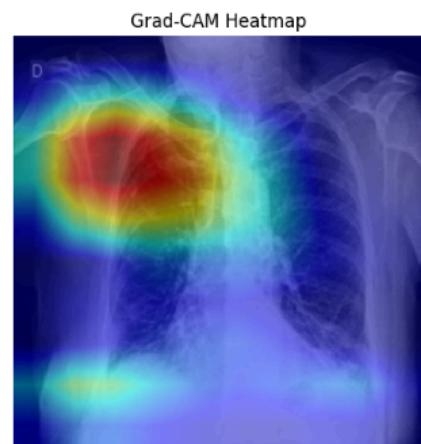
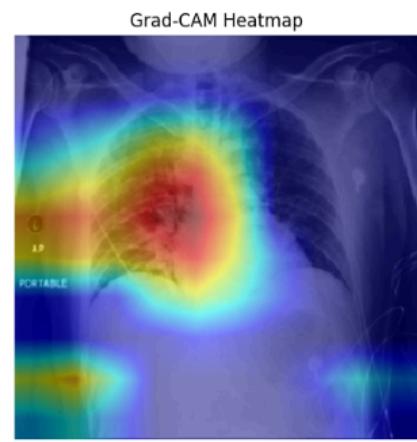


Image 8 - Predicted label: Normal | True label: Normal



On remarque assez rapidement que le modèle se focalise régulièrement sur les annotations inscrites sur les radios. C'est en effet compréhensible.

Si l'on considère que les radios provenant d'un même endroit sont annotées de la même manière, le modèle apprend à reconnaître facilement la provenance des images. Étant donné que dans notre base de données, chaque source ne présente pas les mêmes types de patients, le modèle peut faire le parallèle entre source et type d'affection du poumon.

C'est la raison pour laquelle il est intéressant d'avoir un modèle entraîné uniquement sur les poumons, via les masques. On retrouve le GradCam associé ci dessous

Image 2 - Predicted label: Non COVID | True label: Non COVID

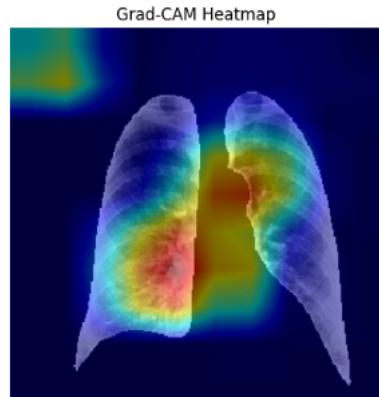
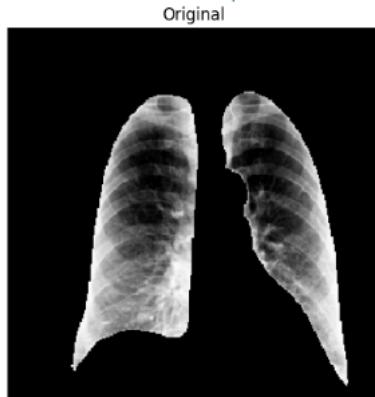


Image 3 - Predicted label: COVID | True label: COVID

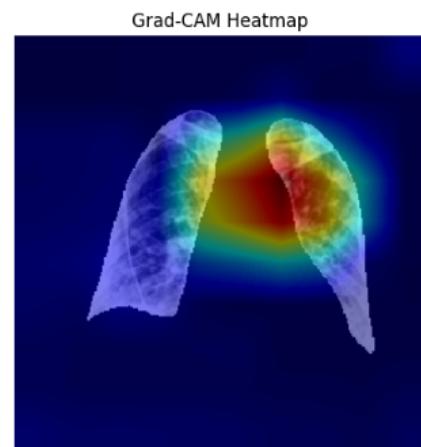
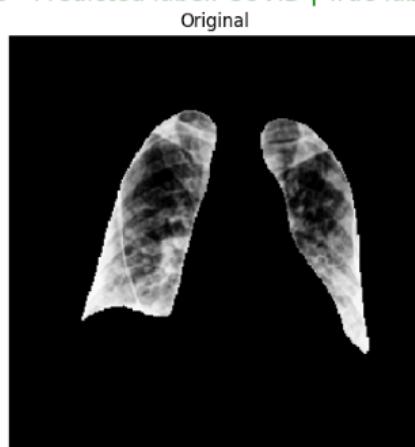
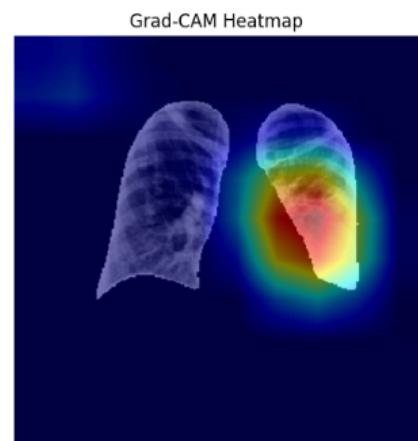
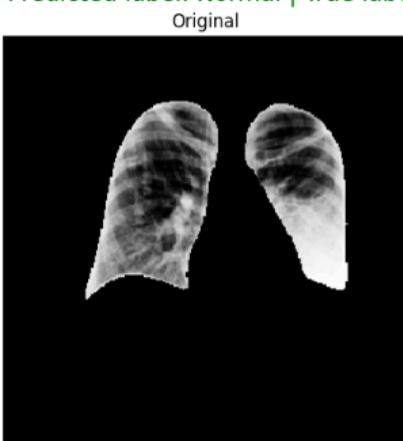


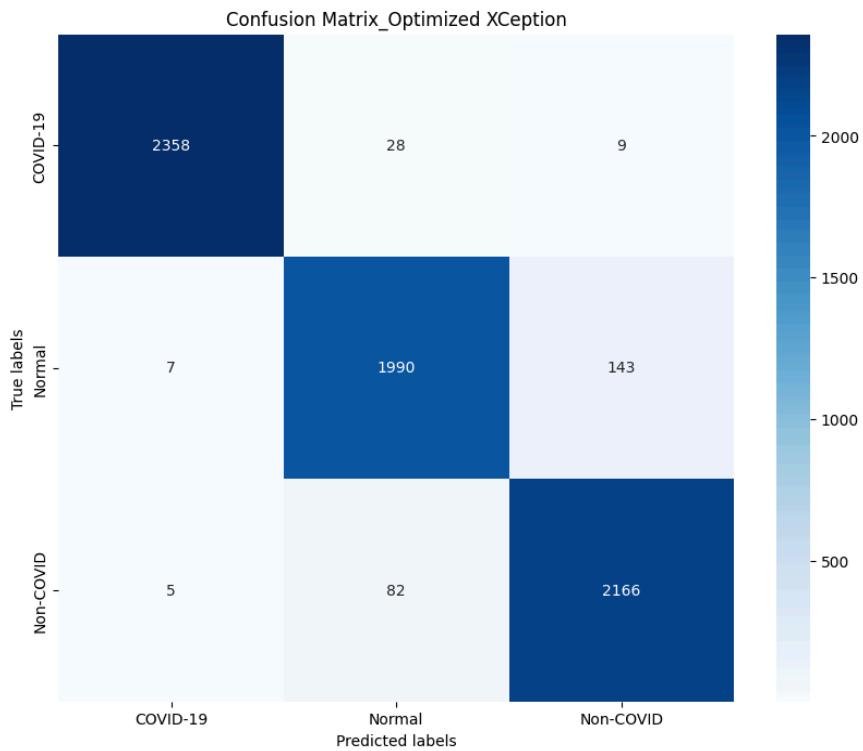
Image 5 - Predicted label: Normal | True label: Normal



On remarque ainsi que ce modèle, bien que moins performant de quelques points, est beaucoup plus scientifiquement acceptable pour détecter une infection pulmonaire, de type COVID ou pas.

5.1.4. Résultats sur le meilleur modèle

Voici les résultats détaillés sur le modèle avec images normalisées par égalisation des histogrammes (Pas d'application de masques).



Rapport de classification

	Precision	Recall	f1-score	Support
COVID-19	0.99	0.98	0.99	2395
Normal	0.95	0.93	0.94	2140
Non-COVID	0.93	0.96	0.95	2253
Accuracy			0.96	6788
Macro avg	0.96	0.96	0.96	6788
Weighted avg	0.96	0.96	0.96	6788

On voit ici que le modèle est plus efficace pour distinguer les cas de COVID que pour distinguer les poumons sains des poumons ayant une autre infection pulmonaire. C'est plutôt encourageant, puisque le but premier de notre modèle est la détection de COVID.

5.2. Modèle DenseNet201

5.2.1. Description du modèle

Le modèle DenseNet201 a été proposé par Gao Huang et ses collègues en 2017. DenseNet est une architecture de réseaux de neurones qui se distingue par ses connexions denses, où chaque couche reçoit en entrée l'ensemble des sorties de toutes les couches précédentes, ce qui permet une meilleure propagation de l'information et des gradients.

Le DenseNet201 est composé de 713 couches réparties en plusieurs blocs principaux :

- Bloc d'initialisation: 1 couche convolutionnelle + pooling
- Dense Blocks: 4 blocs avec des couches convolutionnelles denses
- Transition Layers: 3 couches de transition (convolution + pooling)
- Le bloc de classification suivant est appliqué:

GlobalAveragePooling2D

Couche dense 256 neurones avec activation ‘relu’

Dropout de 20%

Couche dense de 128 neurones avec activation ‘relu’

Dropout de 20%

Dernière couche Dense de Classification avec 3 neurones pour les 3 classes du modèle, avec une activation ‘softmax’

Ici, les images sont normalisées par égalisation adaptative des histogrammes (CLAHE).

5.2.2. Optimisation du modèle

Dans un premier temps, le modèle a été utilisé pré-entraîné avec les données imagenet. Les couches du modèle ont été gelées pour ne pas les réentraîner.

Les couches finales de classification appliquées sont standard:

- GlobalAveragePooling2D
- Couche dense 1024 neurones avec activation ‘relu’
- Dropout de 20%
- Couche dense de 512 neurones avec activation ‘relu’
- Dropout de 20%
- Dernière couche Dense de Classification avec 3 neurones pour les 3 classes du modèle, avec une activation ‘softmax’

On obtient ainsi une **accuracy de 90%** sur 20 époques sur les images normalisées par CLAHE. La perte est de 0.27. Pour les images masquées, l'**accuracy est de 82%**, et la perte est de 0.44.

Avec déjà de bon résultats, une amélioration du modèle par “fine-tuning” est réalisée. Les couches des deux derniers blocs de convolution sont dégelés et le modèle est entraîné. On obtient ainsi une **accuracy de 95.9%** sur les images non masquées après normalisation par CLAHE, et une perte de 0.15. On obtient ainsi des résultats équivalents au modèle précédent. Cependant, les résultats sont légèrement meilleurs sur les images masquées: On obtient **91.1% d'accuracy** et une perte de 0.28.

5.2.3. Interprétabilité des modèles - GradCam

Afin d'utiliser les algorithmes de GradCam proposés par Keras, les modèles ont été recréés de manière fonctionnelle et non séquentielle. Sur les images non masquées, bien que le GradCam montre une assez bonne interprétabilité du modèle qui se concentre la plupart du temps sur les poumons, on observe également que le modèle se focalise parfois sur des annotations ou autres artefacts. En revanche, avec les masques, le modèle n'est pas distrait par les éléments hors poumon et arrive à prédire de manière très correcte la présence d'un COVID ou d'une autre infection.

Image 4 - Predicted label: COVID | True label: COVID

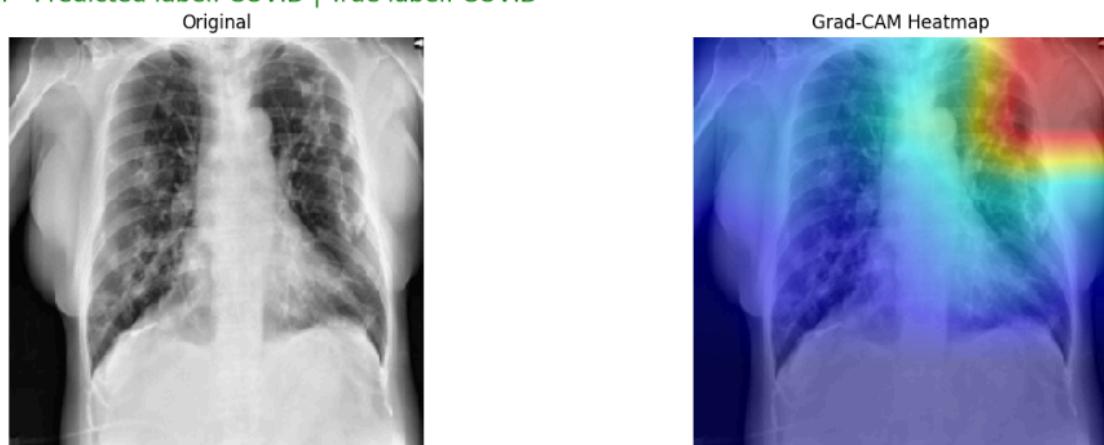


Image 7 - Predicted label: COVID | True label: COVID

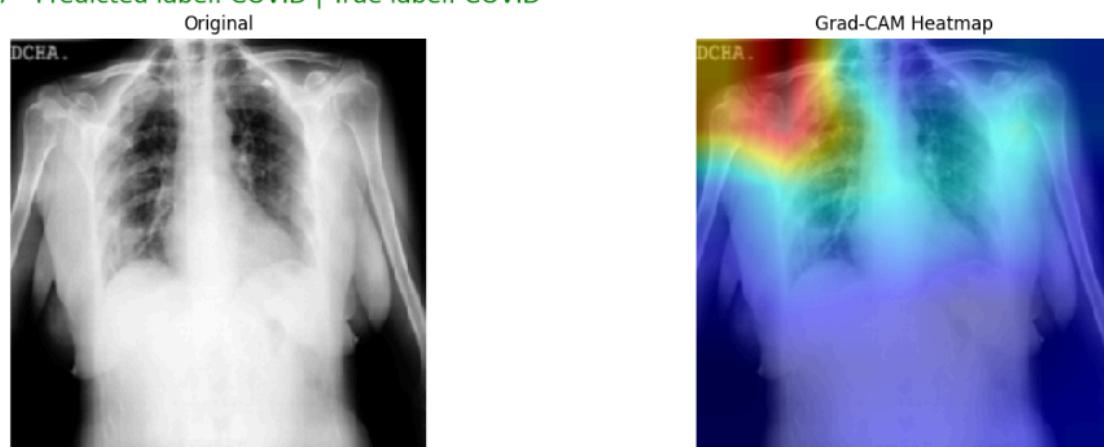


Image 6 - Predicted label: Normal | True label: Normal



Voici quelques exemples sur images masquées :

Image 1 - Predicted label: COVID | True label: COVID

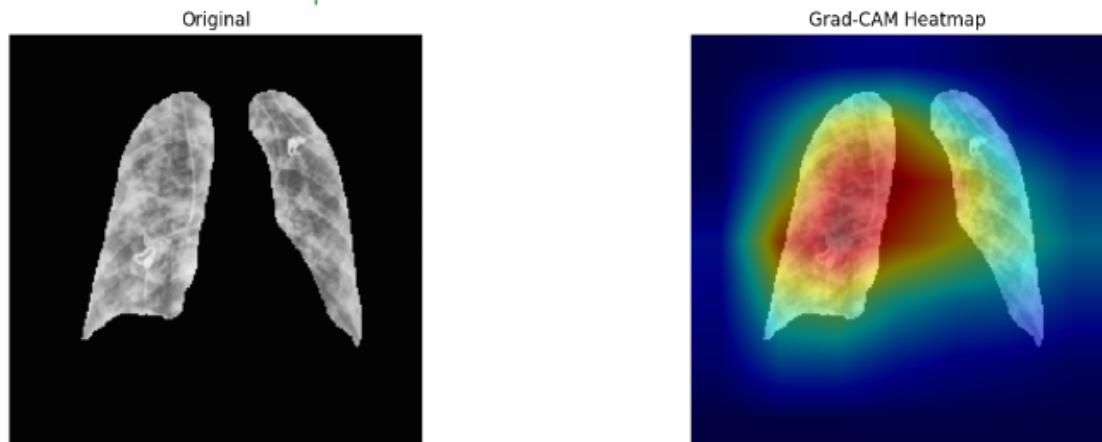


Image 2 - Predicted label: Non COVID | True label: Non COVID

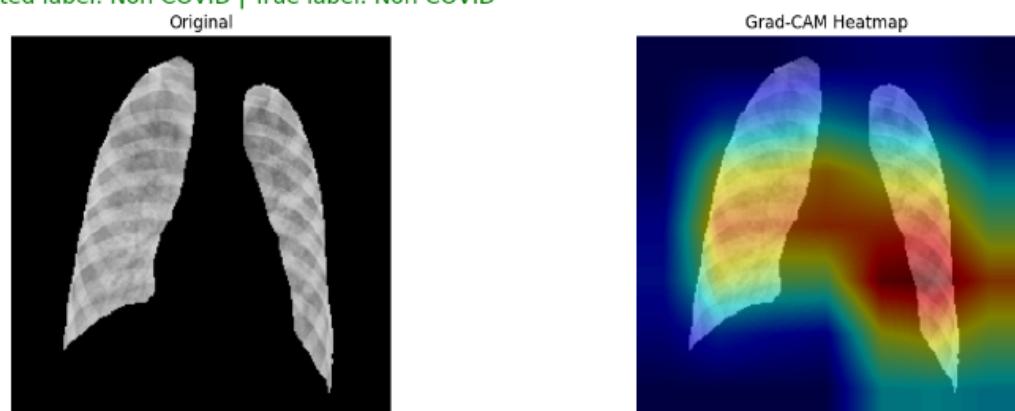
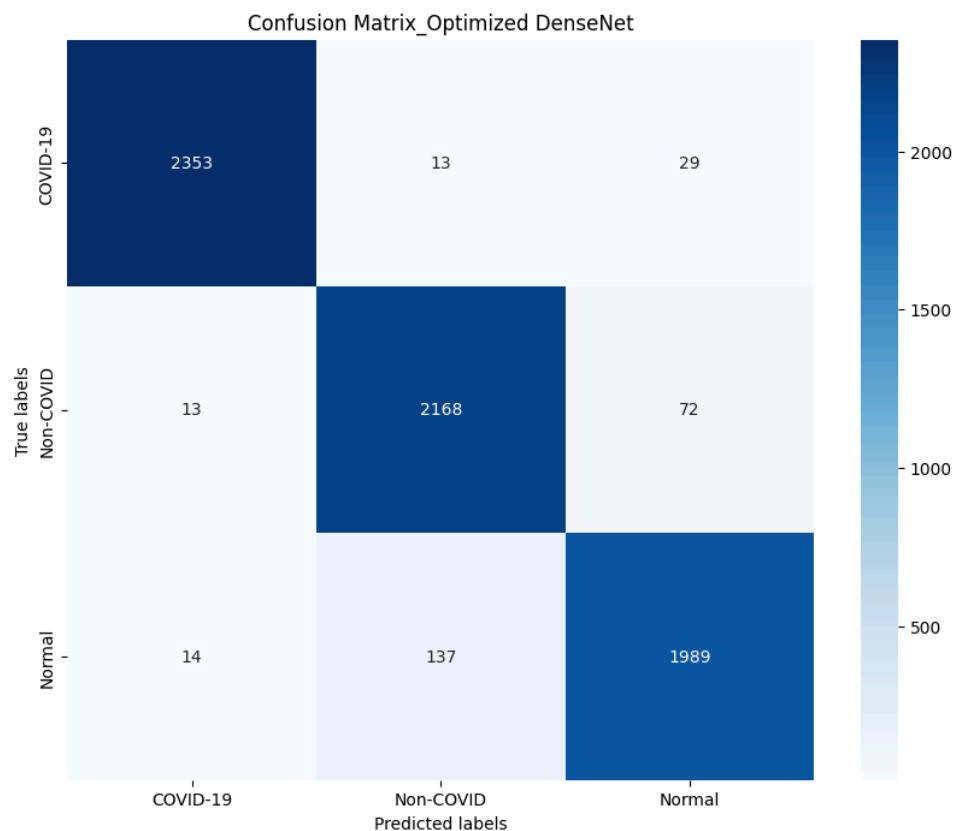


Image 4 - Predicted label: COVID | True label: COVID



5.2.4. Résultats sur le meilleur modèle

Matrice de confusion avec résultats du modèle non masqué:

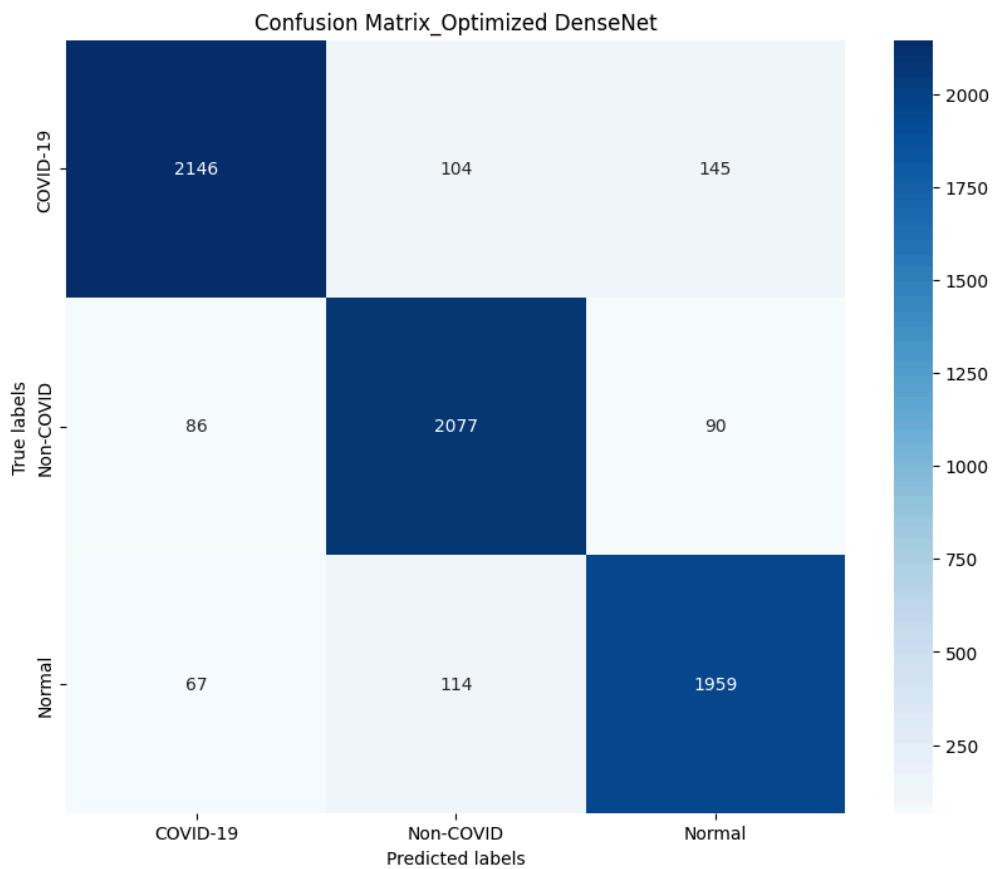


Rapport de classification:

	Precision	Recall	f1-score	Support
COVID-19	0.99	0.98	0.99	2395
Normal	0.95	0.93	0.94	2140
Non-COVID	0.94	0.96	0.95	2253
Accuracy			0.96	6788
Macro avg	0.96	0.96	0.96	6788
Weighted avg	0.96	0.96	0.96	6788

Le rapport est quasiment identique au précédent modèle, avec 1% de précision en plus sur les infections Non-COVID.

Pour les images masquées:



Rapport de classification:

	Precision	Recall	f1-score	Support
COVID-19	0.93	0.90	0.91	2395
Normal	0.89	0.92	0.90	2140
Non-COVID	0.91	0.92	0.91	2253
Accuracy			0.91	6788
Macro avg	0.91	0.91	0.91	6788
Weighted avg	0.91	0.91	0.91	6788

On obtient ainsi des résultats très acceptables sur les images masquées, avec notamment une précision de 93% sur les labels “COVID”, et par contre une précision de seulement 89% sur les labels “Normal”

5.3. MobileNetV2 [22]

Les modèles de type ResNet et DenseNet étant assez longs à tuner et entraîner et nos ressources en GPU étant limitées sur Kaggle ou Colab, nous avons recherché des alternatives potentiellement plus légères (au niveau de l'architecture ou du nombre de paramètres), pour déterminer si le gain de temps annoncé dans la littérature au niveau de l'utilisation du modèle pouvait se retrouver aussi sur l'entraînement.

MobileNetV2 nous semblait un bon candidat car il a été conçu spécialement pour prendre en compte des contraintes de ressources limitées, dans le contexte d'appareils mobiles ou embarqués, comparativement par exemple à un DenseNet121 (non présenté ici mais très proche du DenseNet201) qui est un modèle plus lourd, il comporte 3,6 millions de paramètres vs 8,3 millions, et 53 couches vs 121.

MobileNetV2 est disponible dans TensorFlow avec la méthode MobileNetV2.

5.3.1. Description du modèle

Les caractéristiques de MobileNetV2 sont :

- des **convolutions séparables en profondeur** au lieu de couches de convolutions standards : elles permettent de découper une couche standard en 2 couches plus simples et moins coûteuses en calcul et en paramètres : il y a d'abord, une convolution "depthwise" qui applique un filtre unique à chaque canal d'entrée, suivie par une convolution "pointwise" (1x1) qui combine linéairement les sorties de la première étape pour créer de nouvelles features.
- des **bottlenecks linéaires** : les couches de type bottleneck sont des couches intermédiaires qui réduisent la dimensionnalité avant de la ré-étendre, ce qui permet quand cette réduction de la dimensionnalité est associée à une transformation de type ReLU, de conserver l'information tout en réduisant la quantité de calculs dans les couches intermédiaires.
- des **blocs résiduels inversés** : Les blocs résiduels connectent le début et la fin d'un bloc de convolution. Contrairement aux résidus classiques où les informations de l'entrée sont ajoutées à la sortie après une série de transformations (comme dans ResNet), dans les résidus inversés l'entrée passe d'abord par une expansion (augmentation de la dimensionnalité), puis par une convolution depthwise, et enfin par une réduction de la dimensionnalité. Cela permet de mieux capturer et transporter les informations importantes tout au long du réseau tout en restant efficace en termes de calculs.
- un nombre de couches réduits : 53 couches organisées de la manière suivante :
 - 1 couche de convolution initiale : une seule convolution standard 3x3 avec 32 filtres, suivie d'une couche de normalisation batch et d'une activation ReLU6.
 - 19 blocs de type bottleneck, chacun constitué de plusieurs couches
 - couches finales :

- convolution 1x1 avec 1280 filtres, suivie d'une couche de normalisation batch et d'une activation ReLU6.
- couche de pooling global.
- couche dense pour la classification finale.
- **Taille des kernels et impact sur l'interprétabilité avec Grad-CAM** : les couches de convolutions profondes de MobileNetV2 ont une taille de 3x3, ce qui devrait permettre de capter des détails plus fins que des kernels plus grands de 7x7 comme dans le DenseNet par exemple.
- Entrainement : MobileNetV2 a été entraîné sur ImageNet, les poids sont disponibles dans TensorFlow.
- **Limites** : il y a tout de même une perte d'information, par design. Depuis la publication de MobileNetV2 en 2018, des évolutions de cet algorithme ont été publiées, qui permettent notamment une meilleure accuracy, par ex. Zhao et al., 2022 [23].

5.3.2. Optimisation du modèle

Le processus global a été le suivant :

- Les images sont normalisées par la méthode CLAHE. On a utilisé deux jeux de données: d'un côté les images après normalisation, de l'autre ces mêmes images sur lesquelles sont appliqués les masques correspondants.
- **Création d'un data generator** qui permet aussi de faire le **pré-processing** adapté au modèle :
 - redimensionnement des images en 224x224,
 - division par 255 des valeurs des pixels pour les faire passer entre 0 et 1,
 - avec des batchs de taille 32,
 - et en mélangeant les données à chaque époque pendant l'entraînement pour réduire l'overfitting (paramètre shuffle=True), et sans les mélanger pour la validation et le test.
- **Construction du modèle** :
 - définition de MobileNetV2 comme le “base model”, et en gelant ses couches pour ré-utiliser les features apprises sur le jeu de données ImageNet,
 - ajout de nouvelles couches : Pooling, Dense, Dropout, Dense,
 - utilisation d'un optimizer (Adam), d'un learning rate, d'une fonction de perte (sparse categorical crossentropy) et d'une métrique (accuracy).
- **Tuning** :

Nous avons utilisé un keras tuner pour le choix des hyperparamètres pour les couches ajoutées au modèles de base, ainsi que pour le learning rate.

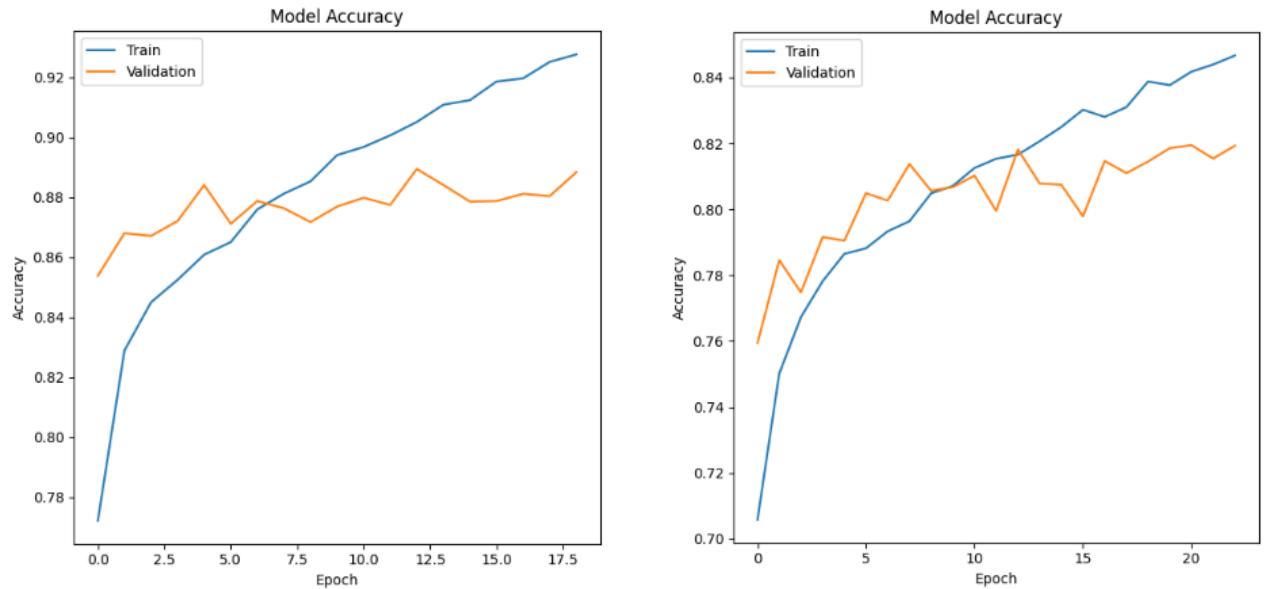
Un tuner de type “BayesianOptimization” a été employé, qui permet de réutiliser les informations des premiers essais dans les suivants, sur 20 essais de 25 époques, avec les informations des 14 premiers essais utilisées pour réaliser les 6 derniers.

Les tuning a été fait uniquement sur les images masquées, en raison des limites de temps GPU disponible, et les meilleurs hyperparamètres ont été utilisés aussi pour les images non masquées.

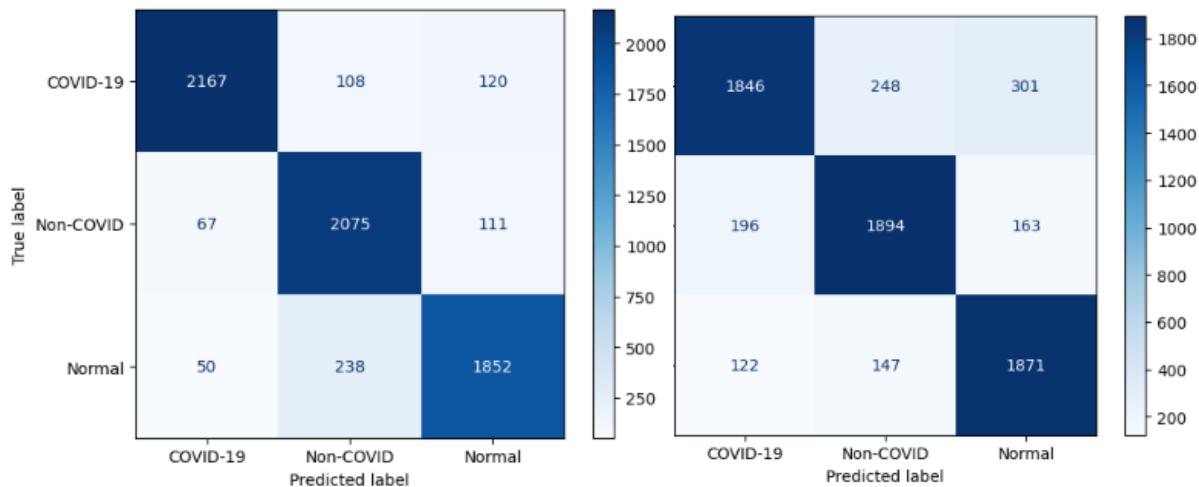
- Définition d'un **callback** de type “early stopping” qui interrompt le tuning au bout d'un certain nombre d'époques durant laquelle la métrique choisie (“val_loss” dans notre cas) ne s'améliore plus. Une option permet de garder les hyperparamètres de l'époque avec la meilleure valeur pour la métrique.
- **Entrainement du modèle avec les meilleurs hyperparamètres :**
 - les meilleurs hyperparamètres étaient :
 - Nombre de neurones pour la couche dense : 384
 - Dropout : 0.4
 - Learning rate : 0.00049
 - création d'un nouveau data generator avec de l'**augmentation de données**, pour tenter de réduire l'overfitting observé le 1er entraînement (rotations, zoom, déformations, retournements)
 - **entraînement** du modèle sur 25 époque avec un callback de type “early stopping”
 - **évaluation** du modèle avec :
 - valeurs d'accuracy et de perte
 - représentation graphique de l'évolution de l'accuracy et de la perte du modèle durant d'entraînement avec les époques
 - matrice de confusion,
 - rapport de classification.
- **Fine tuning** du modèle entraîné :
Le fine tuning permet de dégeler le dernier bloc de couches du “base model”, pour l'entraîner et l'adapter spécifiquement à nos données, il est au courant aussi de réduire le learning rate, pour affiner un peu plus la convergence du modèle.
 - le bloc “block_16_project” a été dégelé et un learning rate de 0.00001 a été utilisé,
 - le fine tuning a été réalisé sur 25 époques, avec un callback de type “stop early”,
 - le modèle est ensuite évalué de manière identique à la phase d'entraînement.

5.3.3. Résultats

Le modèle de base obtient une accuracy de 90% sur les images normalisées, et de 83% sur les images masquées. On peut voir l'évolution sur les graphiques d'entraînement ci-dessous (Images masquées à droite). On y observe déjà un overfitting assez prononcé.



Et les matrices de confusion associées (Images masquées à droite)



Rapports de classification

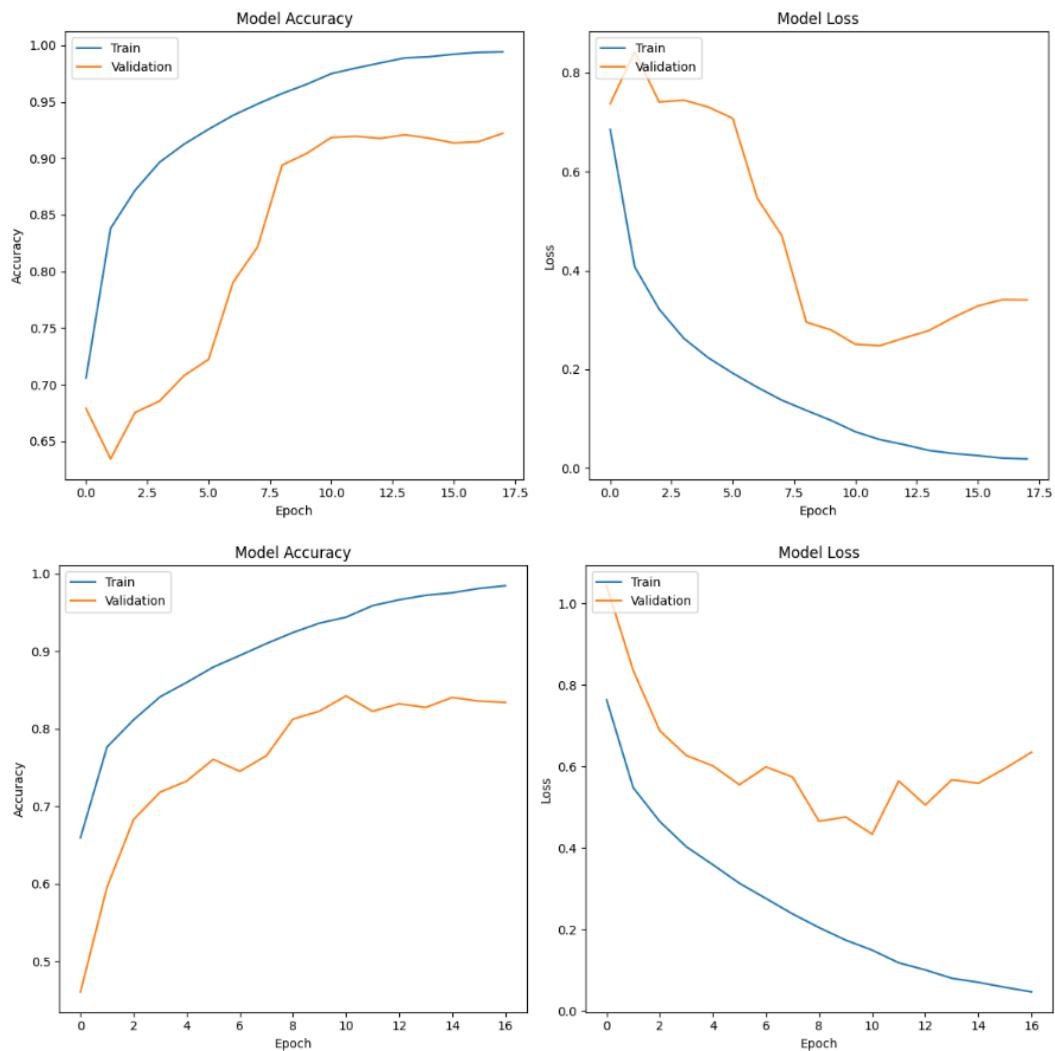
- images masquées :

	Precision	Recall	f1-score	Support
COVID-19	0.85	0.77	0.81	2395
Non-COVID	0.83	0.84	0.83	2253
Normal	0.80	0.87	0.84	2140
Accuracy			0.83	6788
Macro avg	0.83	0.83	0.83	6788
Weighted avg	0.83	0.83	0.83	6788

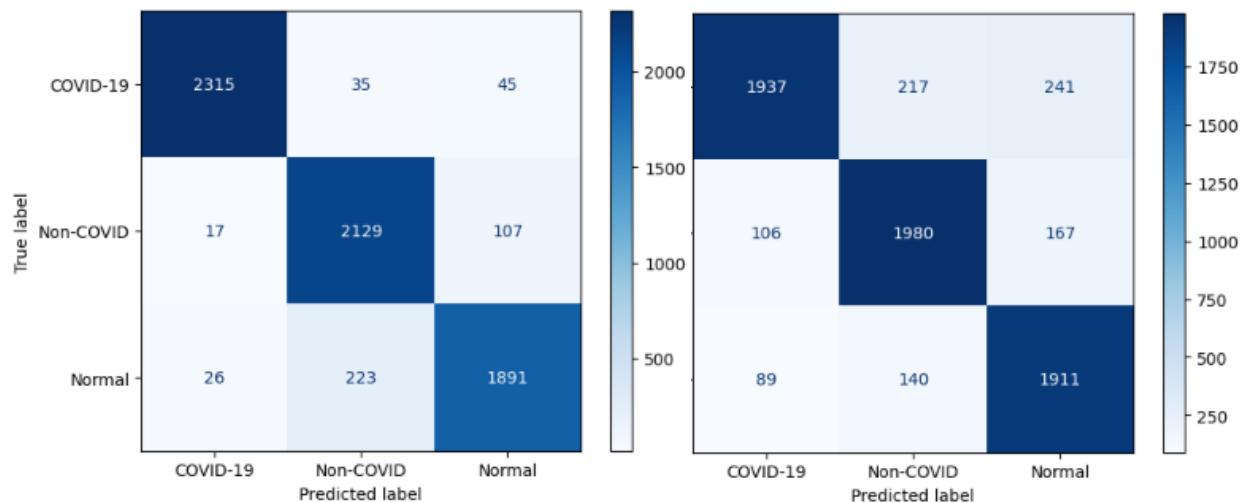
- images non masquées :

	Precision	Recall	f1-score	Support
COVID-19	0.95	0.90	0.93	2395
Non-COVID	0.86	0.92	0.89	2253
Normal	0.89	0.87	0.88	2140
Accuracy			0.90	6788
Macro avg	0.90	0.90	0.90	6788
Weighted avg	0.90	0.90	0.90	6788

Après fine tuning, le modèle avec images masquées obtient une accuracy de 85% et une perte de 0.42, là où le modèle avec images seulement normalisées obtient une accuracy de 93% et une perte de 0.19. On observe à nouveau un overfitting prononcé, et ce dès le début de l'entraînement. Ci-dessous les courbes d'entraînement pour les images normalisées d'abord, puis pour les images également masquées.



Et les matrices de confusion associées (Images masquées à droite). On remarque que le modèle avec images masquées, bien que moins performant, détecte tout de même mieux les cas sains (label “Normal”) que l’autre modèle.



Rapports de classification

- images masquées :

	Precision	Recall	f1-score	Support
COVID-19	0.91	0.81	0.86	2395
Non-COVID	0.85	0.88	0.86	2253
Normal	0.82	0.89	0.86	2140
Accuracy			0.86	6788
Macro avg	0.86	0.86	0.86	6788
Weighted avg	0.86	0.86	0.86	6788

- images non masquées :

	Precision	Recall	f1-score	Support
COVID-19	0.98	0.97	0.97	2395
Non-COVID	0.89	0.94	0.92	2253
Normal	0.93	0.88	0.90	2140
Accuracy			0.93	6788
Macro avg	0.93	0.93	0.93	6788
Weighted avg	0.93	0.93	0.93	6788

Pour les images masquées et non masquées le **fine tuning** ne permet de gagner respectivement qu'un et trois points d'accuracy globale, toutes les métriques restent plutôt **faibles notamment pour les images masquées**, même si **équilibrées** entre accuracy et recall. Comme pour les autres modèles, on remarque une meilleure performance sur les images non masquées. On remarque aussi un **overfitting** qui intervient rapidement dans l'entraînement.

Par design on attendait un temps d'entraînement plus court comparé par exemple à un DenseNet 121, et potentiellement une convergence plus rapide (atteinte en moins d'époques). La comparaison est faite sur les images masquées:

Modèle de base

- temps moyen par époque : 47,7s
- convergence en : 18 époques (25 programmées)

Pour comparaison sur un DenseNet 121 sur les mêmes données et sur la même plateforme (Kaggle avec GPU P100) : 57,5s par époque et convergence en 25 époques (sur 25 programmées)

Fine tuning

- temps moyen par époque : 69,5s
- convergence en : 11 époques (25 programmées)

Pour comparaison sur un DenseNet 121 sur les mêmes données : 155,3s par époque et convergence en 6 époques

Ces valeurs ont été mesurées sur un seul entraînement, mais le même ordre de grandeur était retrouvé sur les 4 à 5 tentatives d'entraînement qui ont été nécessaires pour chaque modèle.

On semble bien retrouver un **temps par époque inférieur** pour le MobileNetV2 par rapport à un modèle plus complexe comme le DenseNet121, et une **convergence avant fine tuning plus rapide**. Cependant, cela se traduit également par une baisse bien visible des performances, toute métrique confondue.

5.3.4. Interprétabilité

Comme pour les autres modèles, le MobileNetV2 a été réécrit de façon fonctionnelle pour réaliser un GradCam. Plusieurs couches ont été testées pour déterminer celle pour laquelle l'interprétation avait le plus de sens :

- “block_16_project” : cette couche est juste avant la dernière couche convulsive et avant l'étape de global pooling, ce qui en fait un bon candidat pour capturer des informations de haut niveau tout en préservant la structure spatiale,
- “block_13_expand_relu” : cette couche est assez profonde pour avoir capturé des caractéristiques abstraites et complexes, mais suffisamment en amont pour conserver des informations spatiales détaillées,
- “block_16_project_BN” : dernière couche convulsive avec Batch Normalization.

Au final c'est la dernière couche du modèle “**block_16_project_BN**” qui nous montrait les informations les plus pertinentes :

- images masquées :

Image 5 - Predicted label: Non COVID | True label: COVID

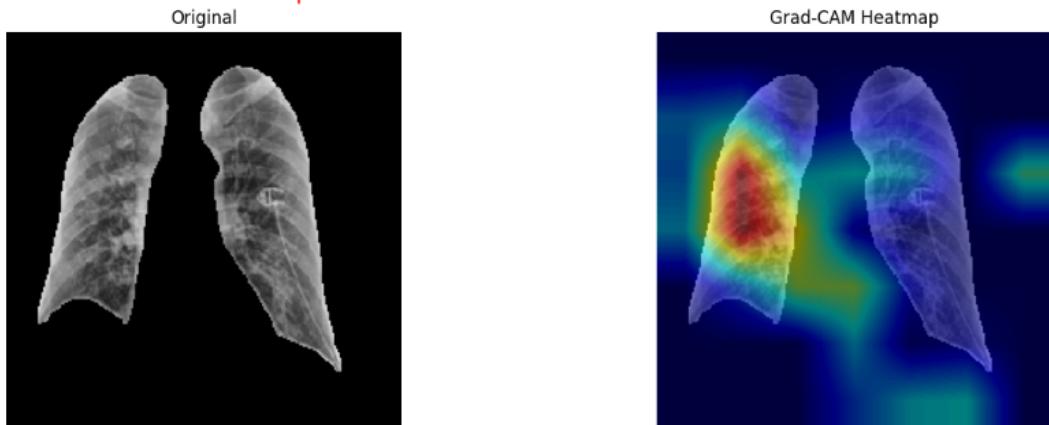


Image 4 - Predicted label: Normal | True label: Normal

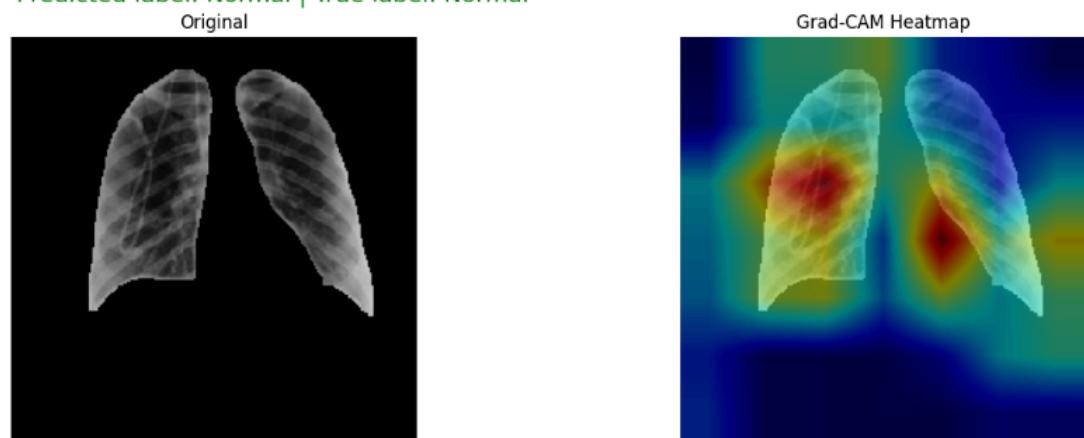
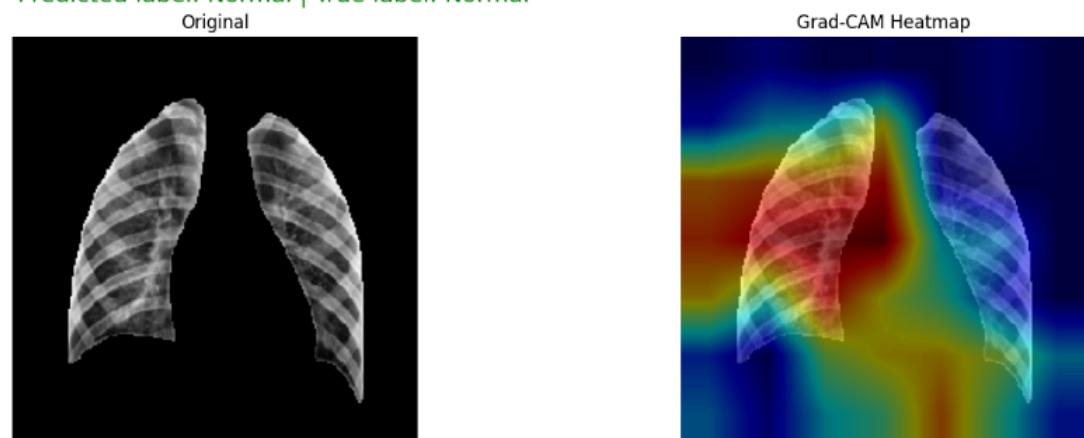


Image 6 - Predicted label: Normal | True label: Normal



- images non masquées :

Image 1 - Predicted label: Normal | True label: Normal

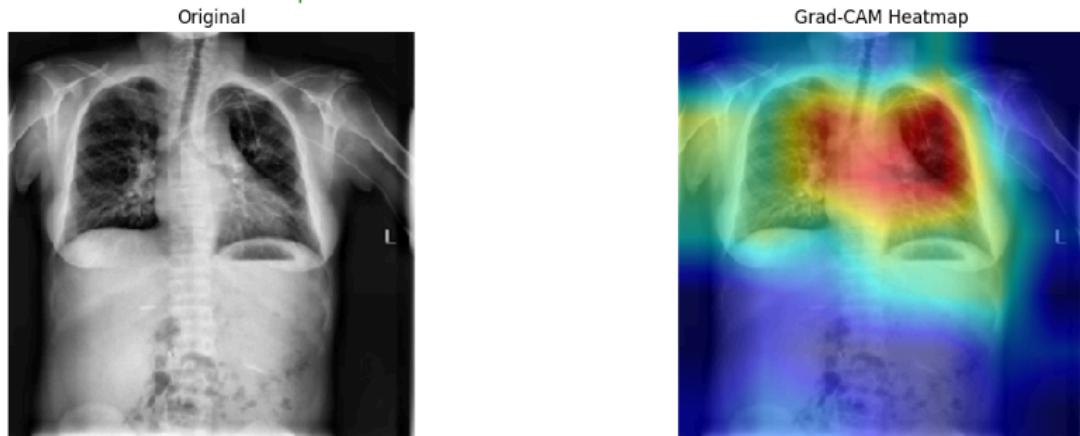
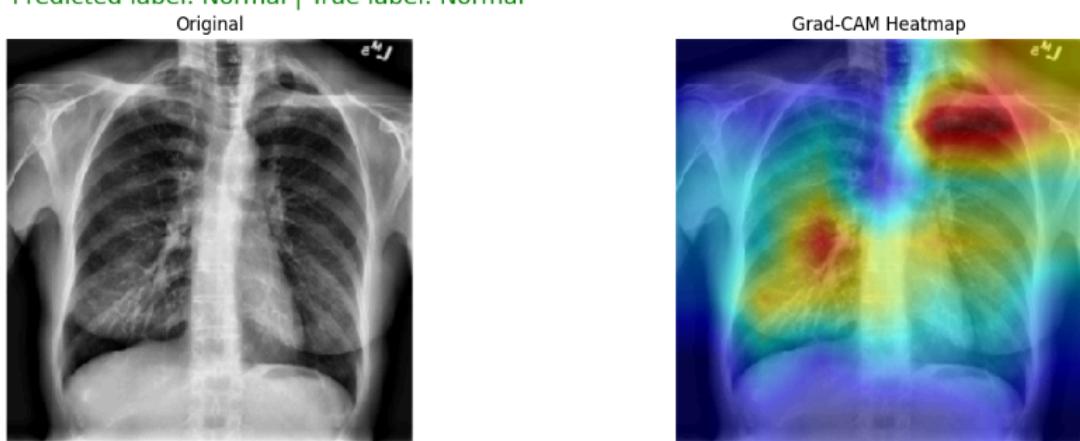


Image 4 - Predicted label: COVID | True label: COVID



Image 8 - Predicted label: Normal | True label: Normal



Avec une taille de kernel de 3x3 on aurait pu s'attendre à des zones d'intérêt plus petites. Globalement le modèle semble se concentrer sur les poumons, ce qui n'est pas toujours le cas avec des performances moyennes comme celles observées pour ce modèle. En comparant plusieurs résultats du Grad-Cam il est **difficile d'observer une tendance précise** sur une zone qui serait particulièrement utilisée pour la prédiction. Sur les images non masquées, on remarque parfois la prise en compte des annotations. Au final malgré un **gain de temps** dû à son architecture plus légère, les **performances moyennes et l'interprétabilité peu claire ne permettent pas de retenir MobileNetV2** pour la suite.

5.4. VGG19

5.4.1. Description du modèle

Le choix de ce modèle vient de la publication “Deep-chest: Multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases” [24] . Les auteurs ont développé des modèles de multi-classification pour le diagnostic du COVID, de la pneumonie et de certains cancers. Les différents scores obtenus (jusqu'à 98% de précision) ont pesé dans le choix de ce modèle, ainsi que sa relative simplicité (peu de couches de neurones).

Le modèle a été développé en 2015, par Karen Simonyan et Andrew Zisserman [25]. L'architecture du modèle est présentée dans la page suivante.

Ce modèle est un réseau de neurones convolutif (CNN) pré-entraîné régulièrement utilisé pour la classification d'images. Il est composé de 19 couches (convolutions et pooling), et il est généralement utilisé comme extracteur de caractéristiques.

- Dans un premier temps, **les couches de convolution (16 premières couches) vont d'abord extraire les caractéristiques locales de l'image d'entrée en appliquant des filtres convolutifs.** En input, on retrouve une image à taille fixe de (224x224), il est donc important de penser à convertir les images que nous souhaitons classer. En étape de pré-processing, on soustrait simplement la valeur moyenne de RGB pixel par pixel. L'image passe ensuite par un stack de couches convolutionnelles avec un kernel de 3x3 (taille la plus petite pour capter les notions de haut/bas/gauche/droite/centre). La stride de convolution est fixée à 1 ; le padding spatial d'une couche de convolution est de telle sorte que la résolution spatiale est conservée après la convolution (padding de 1 pixel pour un kernel 3x3). Ensuite, un pooling est réalisé par 5 couches de Max-Pooling (toutes les couches de convolution n'ont pas forcément un pooling par la suite). Ce Max-Pooling est effectué sur une fenêtre de 2x2 pixels, avec un stride de 2.
- Dans un second temps, **les trois dernières couches qui sont des Fully Connected Layers, vont utiliser les caractéristiques extraites par les couches de convolution pour classifier l'image.** Les deux premières couches contiennent 4096 canaux chacune, et la troisième performe une classification de type ILSVRC sur 1000 canaux. La couche finale est une couche d'activation softmax. Ce modèle a été pré-entraîné sur la base de données ImageNet, qui possède 1000 classes d'images (d'où les 1000 canaux de la dernière couche). Il est important de noter que ce modèle n'a pas été pré-entraîné sur des images médicales, comme des radiographies. Ainsi, pour ne pas interférer dans la classification lors de l'entraînement du modèle, les couches de classification seront freeze avant le fine tuning.
- Toutes les hidden layers du modèle possèdent une rectification nonlinéaire (ReLU). Ces couches permettent de mieux classer les images en éliminant certaines relations linéaires qui fausserait la classification.

A ce modèle VGG19 sont rajoutées **2 couches de convolutions et 2 Fully Connected Layers** dans le but d'adapter celui-ci à notre jeu de données. **Les deux CNN rajoutées poursuivent l'extraction de caractéristiques, et les deux FCL affinent la classification de l'image.**

- Les 2 blocs CNN poursuivent la feature extraction : le premier bloc à une couche de convolution avec une couche ReLU ; le deuxième à 2 couches de convolution, 2 couches de ReLU, une couche de Max-Pooling et enfin une couche de Dropout (qui permet de réduire l'éventuel overfitting).
- L'output de ces couches de feature extraction est ensuite passé à une couche flatten, qui convertit la forme des données en un vecteur unidimensionnel, qui représente la première étape de classification. La partie de classification est constituée d'une couche Dense de 512 neurones suivie d'une autre couche de Dropout. L'output final est produit par une couche Dense de 3 neurones et d'une fonction d'activation Softmax, qui va classer les images en trois catégories : COVID, Non-COVID et Normal.

A noter que ce modèle a été testé en utilisant Pytorch sur un ordinateur personnel, avec une Geforce GTX 1070 comme GPU.

5.4.2. Pré-Processing

Le pré-processing a été réalisé en considérant **le type d'image que nous avions (radiographie), la quantité de données, les pré-requis de pré-processing du VGG19, et la possibilité d'overfitting.**

En parcourant les images, nous nous sommes rendus compte qu'il y avait une grande disparité de qualité des données, avec des images non centrées, une luminosité différente de chaque radio, et des artefacts médicaux (cathéters, patches, écriture sur les radios,...). Nous avons donc d'abord réalisé une **normalisation des pixels de chaque radio en appliquant une normalisation de type equalisation d'histogrammes CLAHE**, adaptée aux images médicales. Nous avons également appliqué la normalisation décrite pour le modèle VGG19. Enfin, une **augmentation des données standard sur le set Train** a été réalisée : rotation aléatoire, flip horizontal aléatoire, redimensionnement, et altération des couleurs).

Le pré-processing est effectué grâce à la méthode '**transform**' de Pytorch.

5.4.3. Optimisation du modèle

Le déroulement de l'optimisation a été empirique, basé sur les meilleurs paramètres que l'on retrouve dans l'article. Le package d'optimisation utilisé dans Pytorch est optuna, qui est similaire à Keras pour Tensorflow.

Les paramètres testés étaient :

- le taux de dropout des deux couches : de 0.4 à 0.7, avec un pas de 1
- le nombre de neurones des deux couches denses : de 128 à 512, avec un pas de 128
- le learning rate : 0.00006, 0.00009 et 0.001

Les meilleurs paramètres retenus sont : 0.5, 0.5, 512, 128 et 0.00006.

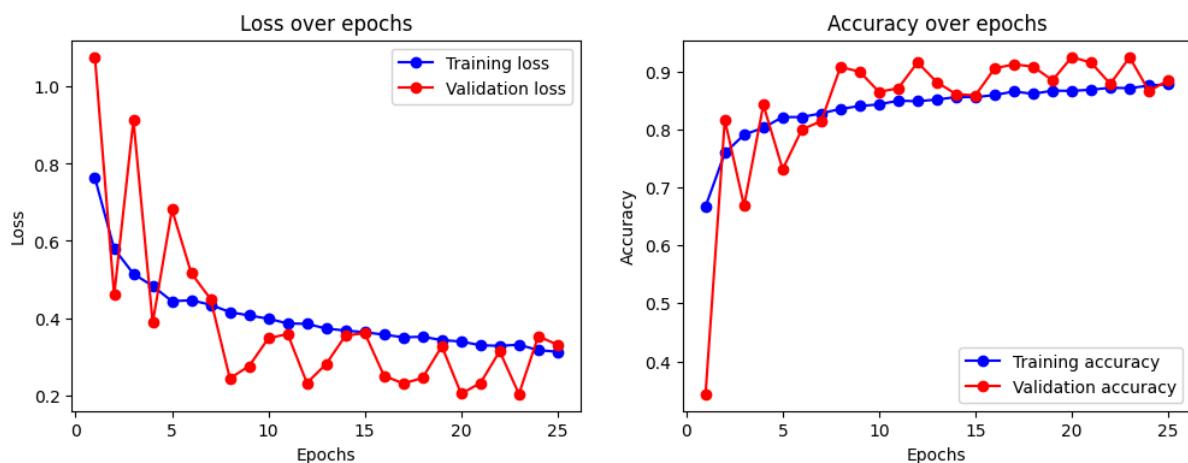
Cette optimisation s'est déroulée sur 10 époques, avec 5 essais. Deux callbacks ont été implémentés pour réduire le temps d'exécution : un early stopping et un checkpoint. Le temps d'exécution était d'environ 5h.

5.4.4. Résultats

Dû à une contrainte de temps en rapport avec le matériel de l'ordinateur personnel, l'étape d'entraînement “brute” avant fine tuning n'a pas été réalisée. Les dernières couches de convolution du modèle ont donc été directement “dégelées” (2 derniers blocks de convolutions). Nous avons réalisé un entraînement sur les images normalisées sans les masques, puis sur les images avec les masques appliqués.

Images normalisées CLAHE

Le modèle est entraîné sur 25 époques, avec un set de Train et un de validation. **Le temps d'entraînement était de : 2h et 30 minutes, avec une moyenne de 6 minutes par époque.** Nous obtenons les courbes de précision et de perte suivantes :



Après entraînement du modèle (Train/Val), nous obtenons une précision de 93.7%, et une perte de 0.18.

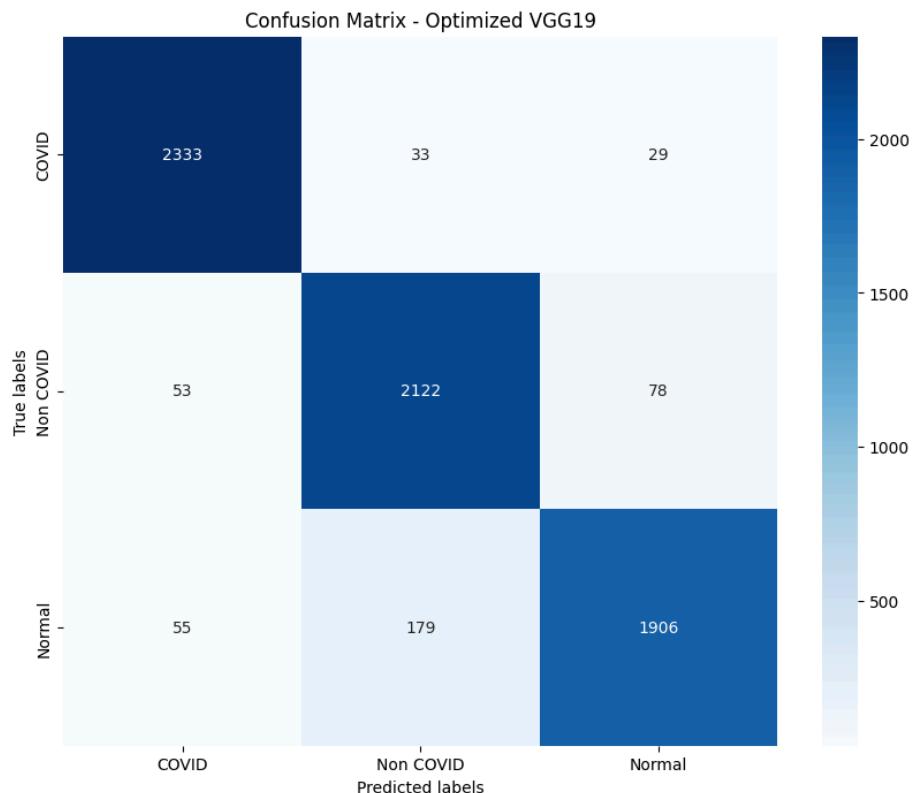
La précision de ce modèle pour les trois classes est satisfaisante (bien qu'inférieure aux modèles précédents, toutefois le recall pour les radios “Normal” est significativement moins élevé (0.89) que pour les autres classes. Le temps d'entraînement est aussi significativement plus long que sur les modèles exécutés avec kaggle (presque 6 fois plus long).

Rapport de classification:

	Précision	Recall	f1-score	Support
COVID	0.96	0.97	0.96	2395
Non COVID	0.91	0.94	0.93	2253
Normal	0.95	0.89	0.92	2140
Accuracy			0.94	6788
Macro avg	0.94	0.94	0.94	6788
Weighted avg	0.94	0.94	0.94	6788

Les courbes de précision et de perte pour le set de validation présentent des variations plus intenses que celles du set train. Cela suggère un overfitting. A noter que cet overfitting était beaucoup plus prononcé dans l'augmentation de données en pré-processing. Il serait également intéressant d'augmenter le nombre d'époques après avoir réduit l'overfitting, afin d'augmenter au maximum les performances du modèle.

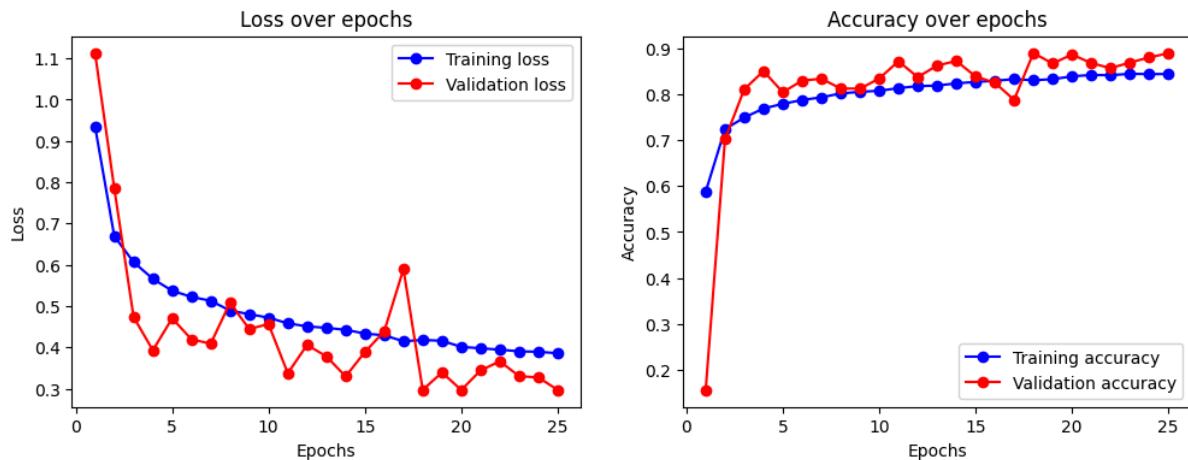
Matrice de confusion:



On constate que le modèle a tendance à attribuer à tort la classe Non COVID à des images COVID. Les affections pulmonaires liées à des afflictions hors COVID présentent peut-être moins de caractéristiques visibles en radio, ce qui trompe le modèle.

Images avec masques appliqués

Le modèle est entraîné sur 25 époques, avec un set de Train et un de validation. **La durée de l'entraînement a été de 2h28 min, avec une moyenne de 5 min et 55 secondes par époque.** Nous obtenons les courbes de précision et de perte suivantes :



Après entraînement du modèle (Train/Val), nous obtenons une accuracy de 91.7%, ainsi qu'une perte de 0.24. :

Rapport de classification :

	Précision	Recall	f1-score	Support
COVID	0.91	0.94	0.93	2395
Non COVID	0.91	0.92	0.92	2253
Normal	0.93	0.88	0.91	2140
Accuracy			0.92	6788
Macro avg	0.92	0.92	0.92	6788
Weighted avg	0.92	0.92	0.92	6788

Les scores obtenus sont presque aussi satisfaisant que pour les images sans masque. On obtient même une meilleure précision sur la catégorie "Normal", avec cependant une baisse de précision sur la catégorie COVID (perte de 0.05). On observe également les mêmes problèmes de variation que sur les images sans masque.

5.4.5. Interprétabilité du modèle

Grad-Cam : **le Grad-Cam n'est pas concluant sur les images sans masque.** On retrouve quand même une majorité des features sur les poumons dans le cas d'une image COVID, mais pour les deux autres classes ce n'est pas vraiment le cas. En revanche, sur les images avec masque, **le Grad-Cam semble être concluant sur les images avec masque**, montrant qu'une majorité des features sur lesquelles se base le modèle pour faire une prédiction semblent bien être des structures sur le poumon.

GRAD-CAM sur images normalisées (CLAHE)

Image 1 - Predicted label: COVID | True label: COVID

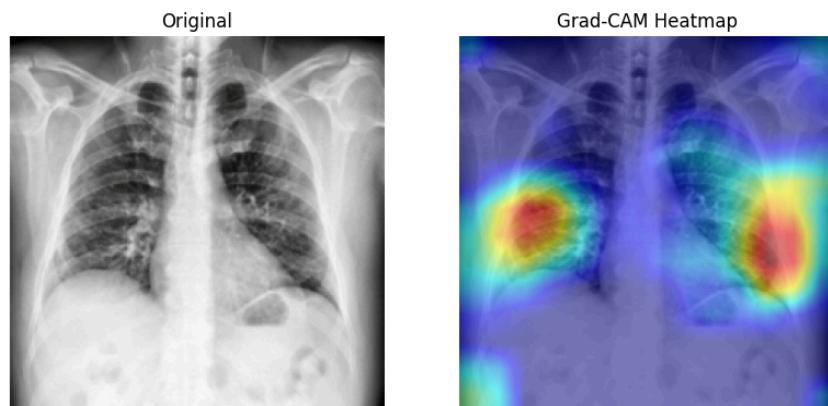


Image 5 - Predicted label: Non COVID | True label: Non COVID

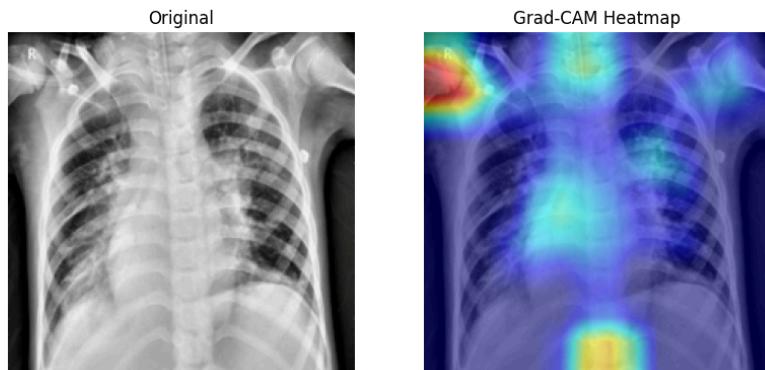
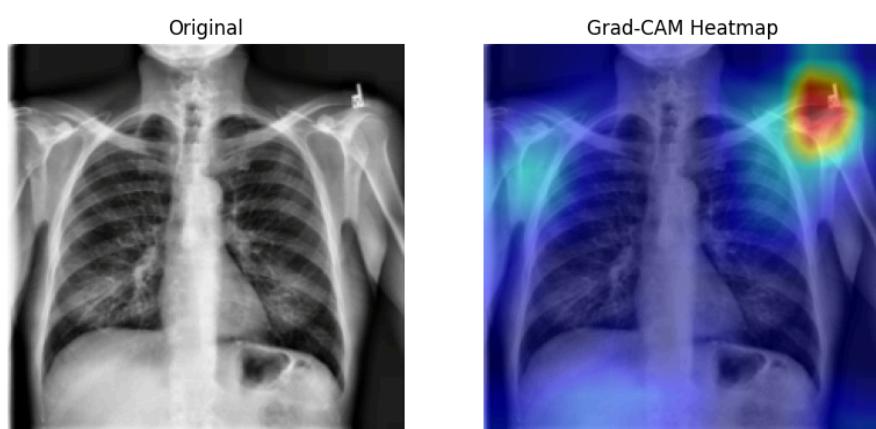


Image 7 - Predicted label: Normal | True label: Normal



GRAD-CAM sur masques appliqués

Image 1 - Predicted label: COVID | True label: COVID

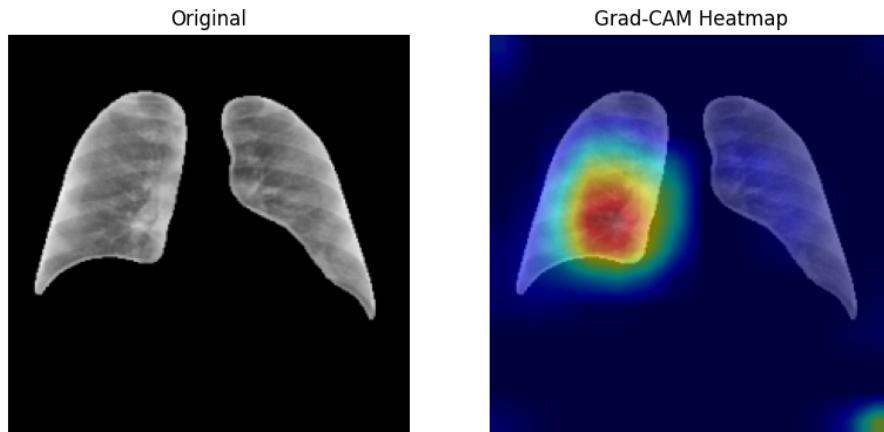


Image 7 - Predicted label: Normal | True label: COVID

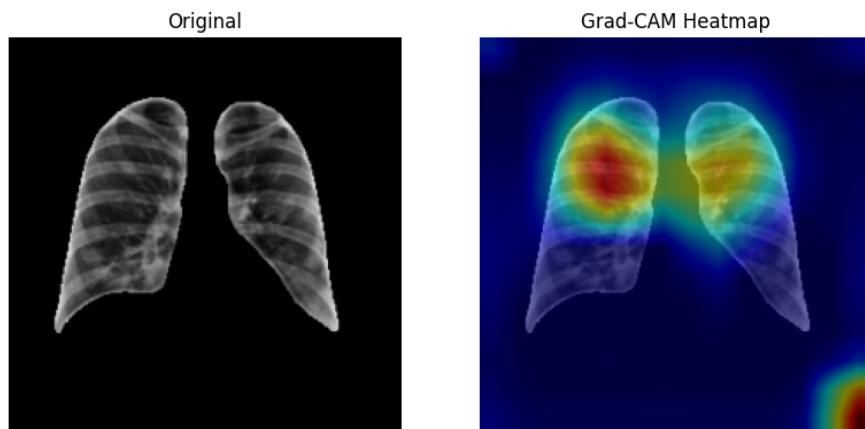
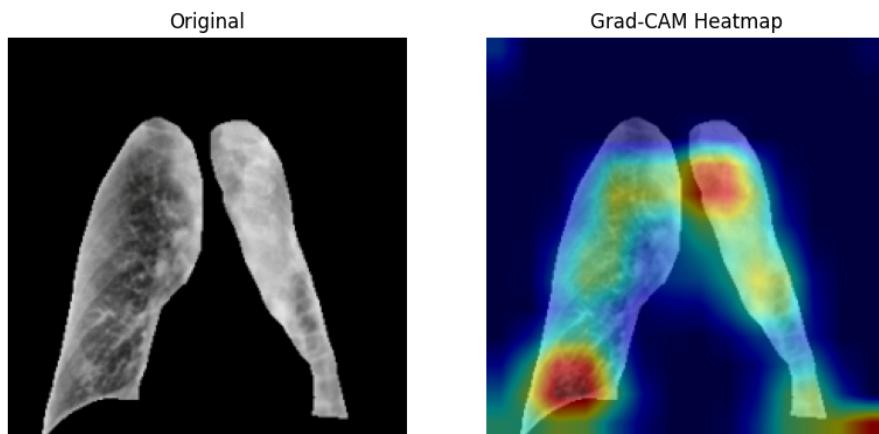


Image 9 - Predicted label: COVID | True label: COVID



Le GradCam est plus satisfaisant sur ce modèle, bien qu'on observe systématiquement un artefact en bas à droite des images, dans une zone qui est masquée en permanence.

5.5. Conclusion sur les modèles

5.5.1. Performances

Modèle	Données	1er entraînement		Fine Tuning		Limites
		époques	Métriques : accuracy	époques	Métriques : accuracy	
Xception 2017, 132 couches	images normalisées (HE)	10	89.1%	20	96.1%	Interprétabilité : focus parfois sur des annotations
	images normalisées HE et masquées			20	90.3%	Overfitting prononcé
DenseNet201 2017, 713 couches	images normalisées CLAHE	30	90%	30	95.9%	interprétabilité : focus parfois sur des annotations
	images normalisées CLAHE et masquées	30	82%	30	91.1%	
MobileNetV2 2018, 53 couches	images normalisées CLAHE (hyperparamètres sur images masquées)	25	89.7%	25	93.3%	overfitting, interprétabilité : focus sur des zones trop larges et parfois sur des annotations
	images normalisées CLAHE et masquées	25	82.6%	25	85.8%	overfitting, interprétabilité : focus sur des zones trop larges
VGG19 2015, 19 couches	images normalisées CLAHE			25	93.7%	Léger overfitting, interprétabilité : l'algorithme utilise parfois des zones extérieures aux poumons
	images normalisées CLAHE et masquées			25	91.6%	Léger overfitting

Au vu des performances et des limites, c'est le **DenseNet201** qui est retenu. Le modèle obtient en effet une accuracy presque au niveau du modèle Xception (à 0.2% près) sur les images standard, mais avec une absence totale d'overfitting, même à 30 époques. Sur les images masquées, il obtient une accuracy très proche du modèle VGG19, toujours sans overfitting, et en étant bien mieux interprétable que ce dernier.

5.5.2. Améliorations

Nous retenons plusieurs axes pour **améliorer les performances de nos modèles**, même si suivant les modèles ou même suivant le jeu d'entraînement nous n'aboutissons pas toujours au même niveau d'amélioration des performances.

- **augmenter les taux de DropOut** pour notamment réduire la complexité du modèle en désactivant des neurones, et donc potentiellement réduire l'overfitting,
- **ajout de callbacks** durant l'entraînement : pour arrêter l'entraînement quand la métrique ne s'améliore plus, ou pour diminuer le learning rate une fois un plateau atteint,
- **augmentation du jeu d'entraînement** en introduisant des transformations, dans le but d'augmenter la généralisation du modèle,
- sélection des **hyperparamètres** avec un **tuner** : nous avons remarqué d'importantes différences dans les performances d'un même modèle suivant l'exhaustivité de la recherche des meilleures valeurs des hyperparamètres. L'augmentation du nombre d'époques ainsi que du nombre d'essais de combinaisons d'hyperparamètres lors du tuning seraient pertinents pour obtenir des modèles plus efficaces.

Il y a également d'autres pistes qui auraient pu être poussées:

- L'ajout de couches de régularisations l1 et l2 pour diminuer l'overfitting.
- La taille des batchs n'a jamais été optimisée. Elle a été fixée à 32 depuis le début. Si la mémoire nécessaire pour augmenter la taille des batchs le permet, il serait intéressant de voir si cela améliore nos modèles. Il existe notamment une fonction autotune sur tensorflow (encore expérimentale) qui donne le batch_size optimal pour un modèle.
- Nous avons utilisé les modèles de transfer learning disponibles nativement dans Keras, cependant avec un peu plus de temps il aurait été possible d'utiliser d'autres modèles, comme par exemple le CheXNet.
- Enfin, comme dans tous les projets de Machine Learning et Deep Learning, les derniers % d'accuracy peuvent être atteints avec une base de données propre. Cela passe par exemple par le retrait d'annotations, ou encore par une variété plus importante de cas par source (Normal, COVID, Non-COVID), pour justement réduire les biais dus aux annotations, qui sont certainement liées au lieu où la radio a été faite.

Au niveau de l'**interprétabilité**, l'apport des masques apparaît comme pertinent car il permet d'orienter fortement l'algorithme vers les poumons, cependant les performances restent en dessous des modèles sur les images non masquées.

Il y a une **limite** également liée à la création des masques : nous avons utilisés ceux présents dans le jeu de données, mais pour une application complète d'un modèle entraîné sur des images masquées, il faudrait ajouter une étape de création et de superposition d'un masque spécifique à chaque image que le modèle aurait à classer.

6. Conclusion

L'application du Deep Learning dans le cadre de la recherche médicale est une des utilisations les plus prometteuses de Machine Learning. Que ce soit dans le domaine pharmaceutique, comme par exemple la résolution du problème de repliement des molécules [26] , de la cinétique médicamenteuse [27] , ou encore comme nous l'avons vu ici dans le diagnostic de certaines maladies pulmonaires. Dans certains cas, les modèles sont même plus performants que les médecins dans le diagnostic [28] .

Dans ce projet, nous avons pu mettre en place un grand nombre de modèles (Xception, MobileNetV2, DenseNET201, VGG19,...) qui se sont tous révélés performants (accuracy de 85,8 % à 96,1 %). Ces modèles ont été entraînés sur un grand nombre de données (environ 30000 images) réparties de façon équilibrée, données qui sont de plus en plus disponibles pour l'ajustement de modèles à l'avenir. Nous avons pu mettre en œuvre une grande partie des différents aspects de la formation Datascientest, du nettoyage de données, au pre-processing de celles-ci et enfin à la construction et l'application de modèles de Deep Learning.

Dans un contexte de diagnostic sur patient, il convient de déterminer l'utilité du DL par rapport aux autres tests, comme le test PCR. L'une des métriques les plus importantes dans le milieu médical est la sensibilité (recall), c'est-à-dire la mesure de la proportion de vraies prédictions positives parmi l'ensemble des vrais cas positifs. Dans le modèle que nous avons choisi, le DenseNET201, celle-ci était de 96%, avec une précision de 99 % : par rapport au gold standard de la détection/diagnostic de COVID-19, la RT-PCR, il s'agit de chiffres équivalents [29] . De plus, ce modèle avait une spécificité (vrai négatifs parmi l'ensemble des cas négatifs) de : $4157/(42+4157) = 0.99$ environ, ce qui est encore une fois équivalent à la RT-PCR [29], [30]. Ces résultats sont encourageants, étant donné que le temps de réalisation d'une radio est de 15 min, versus plusieurs heures pour la PCR, et que la disponibilité des kits PCR n'est pas la même dans le monde.

Nous avions également choisi de faire une classification à 3 catégories, pas uniquement COVID-19 vs Non COVID-19, mais COVID-19/Pneumonie Non COVID-19/ Poumons sain (Normal). L'objectif de cette classification était de produire un modèle permettant non seulement de diagnostiquer les patients COVID-19, mais aussi de pouvoir différencier un patient COVID-19 d'un patient atteint d'une pneumonie. Le modèle DenseNET201 construit permet de faire cette distinction, classant les patients atteints d'une pneumonie avec 94 % de précision et un recall de 96%. Les nombres sont similaires pour des radios d'un patient sain : 95% et 93% respectivement. Toutefois, ces nombres sont plus faibles dans le cas des images avec masques appliqués : 93% et 90% pour images COVID-19, 91% et 92 % pour images de pneumonies, 89% et 92% pour des images de poumons sains.

Il y a cependant des limites à ces modèles. Le premier que nous avons déjà abordé précédemment est l'interprétabilité: Les modèles sur les images non masquées réalisent des prédictions via les annotations des radios en majorité, plutôt que sur les poumons mêmes. Cette limite a été adressée par l'usage de masques, mais cela soulève plusieurs problématiques en usage réel:

- Les modèles avec images masquées sont moins performants, mais le Grad-Cam révèle que la majorité des features utilisées sont sur le poumons même. Peut-on considérer comme acceptable un modèle qui se trompe environ 1 fois sur 10 ? C'est évidemment dépendant de l'usage que l'on souhaite en faire. Dans le cas de la détection d'une maladie infectieuse comme ici, on cherche généralement à éviter les faux négatifs, 10% d'erreur n'est donc pas acceptable.
- Il faut pour chaque nouvelle radio créer le masque associé, afin d'isoler les poumons du reste de la radio. C'est faisable par la librairie OpenCV, mais cela ajoute encore une composante que nous ne maîtrisons pas forcément au sein de notre modèle.
- de plus, les features permettant de classifier les images dans les trois catégories sur des images avec masques appliqués sont plus difficiles à détecter, nécessitant un temps d'entraînement (et donc d'époques) plus élevé, avec peut-être une modification de certaines couches du modèle pour réduire le nombre de pixels pour une classification plus précise.

Enfin, nous avons été grandement limité par les moyens que nous avions pour faire tourner les modèles : 30 h par semaine pour kaggle, et même avec un abonnement google colab, le nombre d'unités de calcul n'était pas suffisant pour faire tourner les gros modèles (type DenseNET ou ResNET sur un grand nombre d'époque). L'essai de Pytorch sur un ordinateur personnel a été concluant, mais nous avons dû réduire la marge de tuning et d'entraînement du modèle (moins d'hyperparamètres à tester, sur un moins grand nombre d'époques...)

7. Bibliographie

- [1] E. B. G. Kana, M. G. Z. Kana, A. F. D. Kana, et R. H. A. Kenfack, « A web-based Diagnostic Tool for COVID-19 Using Machine Learning on Chest Radiographs (CXR) ».
- [2] R. W. Peeling, P. L. Olliaro, D. I. Boeras, et N. Fongwen, « Scaling up COVID-19 rapid antigen tests: promises and challenges », *Lancet Infect. Dis.*, vol. 21, n° 9, p. e290-e295, sept. 2021, doi: 10.1016/S1473-3099(21)00048-7.
- [3] J.-L. He *et al.*, « Diagnostic performance between CT and initial real-time RT-PCR for clinically suspected 2019 coronavirus disease (COVID-19) patients outside Wuhan, China », *Respir. Med.*, vol. 168, p. 105980, juill. 2020, doi: 10.1016/j.rmed.2020.105980.
- [4] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, et U. Rajendra Acharya, « Automated detection of COVID-19 cases using deep neural networks with X-ray images », *Comput. Biol. Med.*, vol. 121, p. 103792, juin 2020, doi: 10.1016/j.combiomed.2020.103792.
- [5] F. Altaf, S. M. S. Islam, N. Akhtar, et N. K. Janjua, « Going Deep in Medical Image Analysis: Concepts, Methods, Challenges, and Future Directions », *IEEE Access*, vol. 7, p. 99540-99572, 2019, doi: 10.1109/ACCESS.2019.2929365.
- [6] F. Xing, Y. Xie, H. Su, F. Liu, et L. Yang, « Deep Learning in Microscopy Image Analysis: A Survey », *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, n° 10, p. 4550-4568, oct. 2018, doi: 10.1109/TNNLS.2017.2766168.
- [7] K. Muhammad, S. Khan, J. D. Ser, et V. H. C. D. Albuquerque, « Deep Learning for Multigrade Brain Tumor Classification in Smart Healthcare Systems: A Prospective Survey », *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, n° 2, p. 507-522, févr. 2021, doi: 10.1109/TNNLS.2020.2995800.
- [8] J. Liu *et al.*, « Applications of deep learning to MRI images: A survey », *Big Data Min. Anal.*, vol. 1, n° 1, p. 1-18, mars 2018, doi: 10.26599/BDMA.2018.9020001.
- [9] P. Seboeck, « Exploiting Epistemic Uncertainty of Anatomy Segmentation for Anomaly Detection in Retinal OCT », *IEEE Trans. Med. IMAGING*, 2019.
- [10] A. M. Tahir *et al.*, « COVID-19 infection localization and severity grading from chest X-ray images », *Comput. Biol. Med.*, vol. 139, p. 105002, déc. 2021, doi: 10.1016/j.combiomed.2021.105002.
- [11] M. E. H. Chowdhury *et al.*, « Can AI Help in Screening Viral and COVID-19 Pneumonia? », *IEEE Access*, vol. 8, p. 132665-132676, 2020, doi: 10.1109/ACCESS.2020.3010287.
- [12] T. Rahman *et al.*, « Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-rays Images », 2020.
- [13] « COVID-QU-Ex Dataset ». doi: 10.34740/kaggle/dsv/3122958.
- [14] A. Degerli *et al.*, « COVID-19 infection map generation and detection from chest X-ray images », *Health Inf. Sci. Syst.*, vol. 9, n° 1, p. 15, déc. 2021, doi: 10.1007/s13755-021-00146-8.
- [15] L. Al Shalabi et Z. Shaaban, « Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix », in *2006 International Conference on Dependability of Computer Systems*, Szklarska Poreba: IEEE, mai 2006, p. 207-214. doi: 10.1109/DEPCOS-RELCOMEX.2006.38.
- [16] S. Roy, K. Bhalla, et R. Patel, « Mathematical analysis of histogram equalization techniques for medical image enhancement: a tutorial from the perspective of data loss », *Multimed. Tools Appl.*, vol. 83, n° 5, p. 14363-14392, févr. 2024, doi: 10.1007/s11042-023-15799-8.
- [17] Y. H. Bhosale et K. S. Patnaik, « Application of Deep Learning Techniques in Diagnosis of Covid-19 (Coronavirus): A Systematic Review », *Neural Process. Lett.*, vol. 55, n° 3, p. 3551-3603, juin 2023, doi: 10.1007/s11063-022-11023-0.
- [18] P. Gifani, A. Shalbaf, et M. Vafaeenezadeh, « Automated detection of COVID-19 using

ensemble of transfer learning with deep convolutional neural network based on CT scans », *Int. J. Comput. Assist. Radiol. Surg.*, vol. 16, n° 1, p. 115-123, janv. 2021, doi: 10.1007/s11548-020-02286-w.

- [19] S. EL-Bana, A. Al-Kabbany, et M. Sharkas, « A Two-Stage Framework for Automated Malignant Pulmonary Nodule Detection in CT Scans », *Diagnostics*, vol. 10, n° 3, p. 131, févr. 2020, doi: 10.3390/diagnostics10030131.
- [20] P. Rajpurkar *et al.*, « CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning ». arXiv, 25 décembre 2017. Consulté le: 29 mai 2024. [En ligne]. Disponible sur: <http://arxiv.org/abs/1711.05225>
- [21] F. Chollet, « Xception: Deep Learning with Depthwise Separable Convolutions ». arXiv, 4 avril 2017. Consulté le: 29 mai 2024. [En ligne]. Disponible sur: <http://arxiv.org/abs/1610.02357>
- [22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, et L.-C. Chen, « MobileNetV2: Inverted Residuals and Linear Bottlenecks ». arXiv, 21 mars 2019. Consulté le: 29 mai 2024. [En ligne]. Disponible sur: <http://arxiv.org/abs/1801.04381>
- [23] L. Zhao, L. Wang, Y. Jia, et Y. Cui, « A lightweight deep neural network with higher accuracy », *PLOS ONE*, vol. 17, n° 8, p. e0271225, août 2022, doi: 10.1371/journal.pone.0271225.
- [24] D. M. Ibrahim, « Deep-chest: Multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases », *Comput. Biol. Med.*, 2021.
- [25] K. Simonyan et A. Zisserman, « Very Deep Convolutional Networks for Large-Scale Image Recognition ». arXiv, 10 avril 2015. Consulté le: 29 mai 2024. [En ligne]. Disponible sur: <http://arxiv.org/abs/1409.1556>
- [26] J. Jumper, « Highly accurate protein structure prediction with AlphaFold ».
- [27] A. Mardt, « VAMPnets for deep learning of molecular kinetics », *Nat. Commun.*, 2018.
- [28] H. Y. F. Wong *et al.*, « Frequency and Distribution of Chest Radiographic Findings in Patients Positive for COVID-19 », *Radiology*, vol. 296, n° 2, p. E72-E78, août 2020, doi: 10.1148/radiol.2020201160.
- [29] M. D. Mair *et al.*, « A systematic review and meta-analysis comparing the diagnostic accuracy of initial RT-PCR and CT scan in suspected COVID-19 patients », *Br. J. Radiol.*, vol. 94, n° 1119, p. 20201039, mars 2021, doi: 10.1259/bjr.20201039.
- [30] R. N. Binny *et al.*, « Sensitivity of Reverse Transcription Polymerase Chain Reaction Tests for Severe Acute Respiratory Syndrome Coronavirus 2 Through Time », *J. Infect. Dis.*, vol. 227, n° 1, p. 9-17, déc. 2022, doi: 10.1093/infdis/jiac317.

8. Environnement de travail

8.1. Ressources

Trois environnements de travail ont été utilisés pour entraîner nos modèles :

- Kaggle : la plateforme nous donne accès à 30h de GPU par semaine
- Google Colab : un certain nombre d'unités de calcul selon la disponibilité dans la version gratuite, ou 90 unités de calcul dans la version payante (ce qui correspondait à entre 10h et 20h d'entraînement)
- PC en local avec un GPU compatible CUDA : GTX1070

La gestion de l'utilisation des ressources en GPU a été primordiale tout au long de la phase de modélisation, nous avons donc utilisé ces 3 environnements pour optimiser le travail, par exemple en faisant le tuning du modèle sur une plateforme et l'entraînement sur une autre.

8.2. Librairies Python

Nous avons travaillé avec TensorFlow sur Kaggle et Colab, pour la majorité de nos modèles. Le travail sur nos machines personnelles s'est limité à la librairie Pytorch pour le VGG19. En effet, l'installation de TensorFlow qui n'est désormais compatible qu'avec Linux nous a causé de nombreux soucis sur nos systèmes Windows. Utiliser une ancienne version de TensorFlow compatible avec Windows nous aurait fait travailler sur des environnements différents, nous avons donc préféré nous concentrer sur les plateformes à distance dans le temps imparti.