

Bootcamp Data Scientist

Février 2024



DataScientest • com

Rapport projet

Analyse de radiographies pulmonaires Covid-19

Alumni :

Alexandre LANGLAIS
Camille RUBI
Chaouki BENZEKRI
Pierre-Jean CORNEJO

Mentor :

Gaël PENESSOT

Table des matières

Résumé.....	3
1. Contexte et objectif.....	4
1.1. Contexte du projet.....	4
1.2. Objectifs.....	4
1.3. Equipe projet.....	5
2. Appréhension des données.....	6
2.1. Exploration des métadonnées.....	6
2.2. Visualisation des données.....	8
2.3. Exploration des radiographies.....	10
2.4. Etude de la surface utile des masques.....	11
2.4. Distribution de l'intensité lumineuse.....	12
3. Preprocessing des images.....	14
3.1. Traitement des images.....	14
3.2. Gestion des labels.....	16
4. Benchmarks et modélisation.....	16
4.1. Etude sur un modèle base-line simple : LeNet-5.....	16
4.2. Nombre d'images optimal.....	18
4.3. Nombre d'époques.....	19
4.4. Usage des masques.....	20
5. Modèles pré-entraînés et transfer learning.....	21
5.1. InceptionResNetV2.....	22
5.2. ResNet121V2.....	24
5.3. DenseNet201.....	25
5.4. VGG16.....	27
5.5. VGG19.....	28
5.6. ConvNextTiny.....	30
5.7. ConvNextBase.....	31
6. Finetuning des modèles sélectionnés.....	32
6.1. Cas des EfficientNet.....	32
6.2. VGG16.....	39
6.3. ResNet.....	41
6.4. DenseNet.....	43
7. Interprétabilité du modèle final.....	47
7.1. Présentation du modèle final sélectionné.....	47
7.2. Interprétabilité.....	47
7.3. Qualités et limites.....	50
Bibliographie.....	54

Résumé

La COVID-19, causée par le coronavirus SARS-CoV-2, peut provoquer des anomalies sévères chez certains patients caractérisée par des lésions pulmonaires spécifiques visibles sur les radiographies sous un effet de “verre dépoli”. Ce projet porte sur la conception d'un système basé sur l'intelligence artificielle, plus spécifiquement grâce à un modèle de réseau de neurones, permettant de détecter la COVID-19 à partir d'une radiographie pulmonaire. Pour ce faire, nous avons à notre disposition un jeu de 21165 radiographies et de masques provenant de huit sources différentes. Nos analyses nous ont permis de mettre en évidence des différences entre les métadonnées fournies et celles des radiographies que nous avons pu prendre en compte lors de notre étape de preprocessing, visant à redimensionner, homogénéiser le nombre de canaux et normaliser les images pour qu'elles soient exploitables dans nos modèles. Ensuite, nous avons pu exploiter l'architecture LeNet-5 afin de tester nos données, l'influence des masques et réaliser plusieurs benchmarks afin de nous donner des pistes sur la manière d'appréhender notre problématique. Nous avons utilisé le *transfer learning* pour tester 15 modèles aux stratégies d'architectures différentes dans le but d'avoir des performances de base élevées selon un protocole de test standardisé. Suite à ces expérimentations itératives et progressives, nous avons sélectionné les quatre modèles avec les meilleurs résultats dans le but de réaliser des ajustements fins : l'EfficientNetB4, le VGG16, le DenseNet201 et le ResNet50. L'étape de *finetuning* a été caractérisée par le dégèlement de couches en profondeur des modèles pré entraînés, l'optimisation des hyperparamètres par keras-tuner et l'ajout de couches supplémentaires pour ajuster les modèles au plus proche de notre problématique initiale. Grâce à ces étapes d'optimisation, nous avons pu atteindre une précision de validation et un F1 Score de 0.96 pour la classe ‘COVID’ et 0.91 pour la classe ‘Normal’ avec le modèle VGG16 finement ajusté et optimisé. Ce modèle à l'avantage d'être à la fois performant sur de nouvelles images et suffisamment léger pour être un parfait compromis entre performance et souplesse d'entraînement.

Mots-clés : Classification d'images ; COVID-19 ; Deep Learning ; DenseNet ; EfficientNet ; Finetuning ; Intelligence Artificielle ; Keras ; Radiographie pulmonaire ; ResNet ; Tensorflow ; Transfer learning ; VGG.

1. Contexte et objectif

1.1. Contexte du projet

La radiographie est une technique d'imagerie médicale largement utilisée à travers le monde pour visualiser les structures internes du corps de manière non invasive. Elle permet en particulier d'observer les tissus osseux et les organes mous. Cette modalité d'imagerie repose sur l'utilisation de rayons X, une forme de rayonnement électromagnétique capable de pénétrer à travers les tissus corporels, mais qui est absorbée à des degrés différents en fonction de la densité et de la composition des tissus traversés.

Les radiographies pulmonaires en particulier, sont couramment utilisées pour évaluer l'état des poumons et des structures avoisinantes. Elles sont précieuses pour le diagnostic et la surveillance de diverses affections pulmonaires, telles que les pneumonies, les tuberculoses, les tumeurs pulmonaires et les maladies respiratoires obstructives. Dans le contexte de la pandémie mondiale de COVID-19, les radiographies pulmonaires sont devenues **un outil essentiel pour le dépistage et le suivi** des patients atteints de cette maladie virale respiratoire.

Le COVID-19, causé par le coronavirus SARS-CoV-2, peut provoquer une pneumonie virale sévère chez certains patients, **caractérisée par des lésions pulmonaires spécifiques visibles sur les radiographies**. Ces anomalies radiographiques comprennent généralement des opacités pulmonaires diffuses, des infiltrats interstitiels et des consolidations alvéolaires. La capacité à identifier et à interpréter ces caractéristiques radiographiques est **cruciale pour le diagnostic rapide et la gestion clinique efficace** des patients atteints de COVID-19.

Dans ce contexte, l'intégration de techniques de Deep Learning pour l'analyse automatisée des radiographies pulmonaires offre un **potentiel prometteur pour améliorer la détection précoce et précise du COVID-19**. Une bonne connaissance des données est essentielle pour planifier efficacement notre démarche.

1.2. Objectifs

L'objectif principal de ce projet est de **développer un système d'analyse automatisée de radiographies pulmonaires pour la détection efficace du COVID-19** sur moins de deux mois en parallèle de notre formation.

En utilisant les techniques de Deep Learning, nous cherchons à créer un modèle de classification capable de **distinguer avec précision les radiographies de patients atteints de COVID-19** de celles présentant d'autres affections pulmonaires ou étant saines. Les objectifs spécifiques incluent l'acquisition et la préparation d'un ensemble de données diversifié et annoté, la conception et l'entraînement d'un modèle de Deep Learning robuste, ainsi que l'évaluation rigoureuse de ses performances en termes de sensibilité, de spécificité et de précision.

Ce projet vise ultimement à fournir un outil de diagnostic complémentaire pour les professionnels de la santé, permettant une détection précoce et précise du COVID-19 à partir de radiographies pulmonaires qualitatives.

1.3. Équipe projet



Alexandre



Chaouki



Camille



Pierre-Jean

Figure. 1 Équipe du projet

Nous réalisons ce projet dans le cadre de notre formation avec DataScientest.

Nous avons choisi ce projet pour la sensibilité éthique qu'il représente sur l'amélioration des diagnostics pour faciliter et accélérer la prise en charge des personnes dans le besoin. Mais aussi sur le caractère technique que représente un tel projet, de la démarche de compréhension d'un projet complexe à l'attente de métriques fiables et rigoureuses en passant par la conception itérative de modèles adaptés à la classification d'images.

Nous n'avons aucune expérience dans le domaine médical, mais le passif de Pierre-Jean et Alexandre dans le secteur de la biologie donne une certaine affinité à la compréhension du domaine.

2. Appréhension des données

2.1. Exploration des métadonnées

Nous avons à notre disposition un important jeu de données provenant de Kaggle.com. Il s'agit de **21165 images de radiographies pulmonaires labellisées dans leur nom, ainsi qu'autant de masques basiques associés aux images**. De plus, quatre fichiers de métadonnées au format *.xlxs sont disponibles en accompagnement des quatre catégories.

Les données sont accessibles sur le site Kaggle à l'adresse suivante :

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

Ce jeu de données comprend quatre catégories de radiographies pulmonaires distinctes :

- **COVID** : Radiographies de patients testés positifs au COVID-19
- **Viral Pneumonia** : Radiographies de patients souffrant de pneumonie virale
- **Lung_Opacity** : Radiographies de patients présentant une infection pulmonaire de diverses natures
- **Normal** : Radiographies de patients en bonne santé

Les quatre fichiers **metadata** fournis semblent propres et cohérents, aucune valeur n'est manquante et il n'y a pas d'erreur concernant les types de variables. Les quatres tables suivent la même organisation :

- **FILE NAME** renvoie vers le nom de l'image
- **FORMAT** renvoie le type de fichier
- **SIZE** renvoie la résolution de l'image
- **URL** renvoie la source de l'image sous forme d'un URL

Nous avons donc concaténé les quatre fichiers pour pouvoir travailler plus facilement sur une table unique grâce à la méthode **.concat()**.

Voici un **.head()** du tableau brut :

	FILE NAME	FORMAT	SIZE	URL
0	COVID-1	PNG	256*256	https://sirm.org/category/senza-categoria/covi...
1	COVID-2	PNG	256*256	https://sirm.org/category/senza-categoria/covi...
2	COVID-3	PNG	256*256	https://sirm.org/category/senza-categoria/covi...
3	COVID-4	PNG	256*256	https://sirm.org/category/senza-categoria/covi...
4	COVID-5	PNG	256*256	https://sirm.org/category/senza-categoria/covi...

Figure 2. Tableau d'origine fourni

En explorant les informations des métadonnées nous avons pu **constater quelques incohérences par rapport aux informations données dans le README.md**. Nous avons donc voulu les confronter aux vraies valeurs extraites des métadonnées des images pour **vérifier la véracité des informations**. Pour ce faire, nous avons utilisé la librairie `os` dans une boucle afin de récupérer plusieurs informations sur chaque radiographie (hors masques). Nous avons donc récupéré :

- **FILE NAME** : le nom du fichier sans son extension.
- **FORMAT** : le type du fichier.
- **SIZE** : sa résolution au format largeur x hauteur
- **LABEL** : grâce à l'intitulé des images sans leur numéro.
- **CHANNELS** : pour avoir l'information du nombre de canaux des images.
- **PATH** : pour avoir le chemin des fichiers et ainsi grandement faciliter les procédures suivantes.

Une fois ces extractions réalisées, nous en avons profité pour réaliser **l'extraction de la source des données** en conservant uniquement le nom de domaine des URLs fournies grâce au package `re` dans une colonne **SOURCE** et **calculer la moyenne de l'intensité lumineuse normalisée** entre 0 et 1 de chaque image comme valeur indicatrice dans une colonne **MEAN_INTENSITY**.

Voici ainsi les 10 premières lignes du tableau obtenu :

	FILE NAME	FORMAT	SIZE	LABEL	CHANNELS	PATH	URL	SOURCE	MEAN_INTENSITY
0	COVID-1	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://sirm.org/category/senza-categoria/covi...	sirm.org	0.572145
1	COVID-10	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://sirm.org/category/senza-categoria/covi...	sirm.org	0.554335
2	COVID-100	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://sirm.org/category/senza-categoria/covi...	sirm.org	0.599524
3	COVID-1000	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://github.com/ieee8023/covid-chestxray-da...	github.com	0.638318
4	COVID-1001	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://github.com/ieee8023/covid-chestxray-da...	github.com	0.509376
5	COVID-1002	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://github.com/ieee8023/covid-chestxray-da...	github.com	0.689172
6	COVID-1003	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://github.com/ieee8023/covid-chestxray-da...	github.com	0.630415
7	COVID-1004	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://github.com/ieee8023/covid-chestxray-da...	github.com	0.531977
8	COVID-1005	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://github.com/ieee8023/covid-chestxray-da...	github.com	0.552627
9	COVID-1006	PNG	299x299	COVID	1	C/Users/Gamy/Desktop/Formations fev24_bootcamp...	https://github.com/ieee8023/covid-chestxray-da...	github.com	0.579442

Figure 3. Tableau des métadonnées réelles

Ainsi, en le comparant au précédent, nous détectons seulement une erreur dans la casse de l'intitulé d'une classe : **Normal** et non **NORMAL** ainsi qu'une erreur vis-à-vis des résolutions : '299x299' et non pas '256x256'. Nous pouvons aussi constater **un nombre différent de canaux**, avec des images qui ne possèdent qu'un seul canal et d'autres avec 3 canaux (ceci ne concerne qu'une partie d'images, 140 images, de la classe **Viral Pneumonia**). C'est une information très importante pour le traitement de nos images pour la suite.

En cherchant les valeurs uniques avec la méthode `.unique()` nous pouvons en tirer des informations :

Les vraies classes uniques sont : ['COVID' 'Lung_Opacity' 'Normal' 'Viral Pneumonia']

Les vraies résolutions uniques sont : ['299x299']

Les vrais formats de fichier uniques sont : ['PNG']

Les vrais nombres de canaux sont : [1 3]

Nombre de radiographies : 21165

Ainsi, chaque image semble bien avoir une résolution homogène ainsi qu'un format de fichier unique. **Nous devrons cependant traiter les nombres de canaux pour n'en garder qu'un seul, comme les images sont en nuances de gris.**

Nous allons maintenant nous concentrer sur les masques à notre disposition. Nous suivons la même procédure que précédemment pour obtenir un tableau `df_masks` qui nous sera bien utile pour la suite de notre étude.

De même, avec la méthode `.unique()` nous observons les paramètres suivants :

Les classes uniques sont : ['COVID' 'Lung_Opacity' 'Normal' 'Viral Pneumonia']

Les résolutions uniques sont : ['256x256']

Les formats de fichier uniques sont : ['PNG']

Les nombres de canaux sont : [3]

Nombre de masques : 21165

Nous constatons ainsi que **les masques sont tous à trois canaux et n'ont pas la même résolution que nos images**. Il sera alors nécessaire de redimensionner nos radiographies et de réduire le nombre de canaux si nous souhaitons les appliquer.

2.2. Visualisation des données

Grâce à la colonne `SOURCE`, nous sommes allés explorer la provenance de nos radiographies.

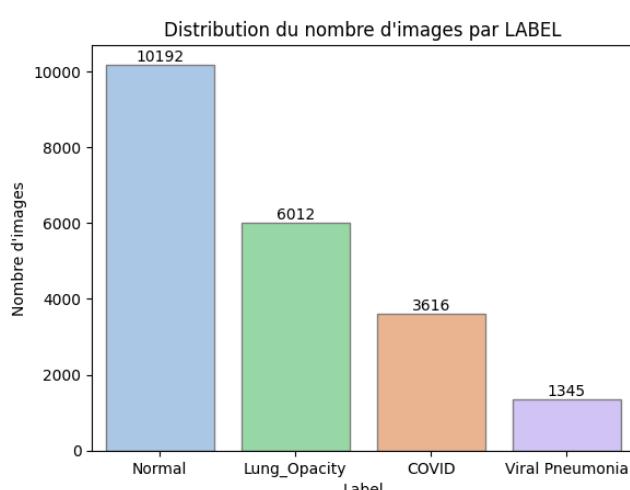


Figure 4. Distribution du nombre d'images par LABEL

La catégorie **Normal** est la catégorie représentée par le plus de radiographies avec plus de 10000 images. A contrario, **Viral Pneumonia** est la catégorie ayant le moins de radiographies avec près de 2000 images. Il est important de garder cette information en tête pour la suite car elle pourrait impacter la domination d'une catégorie sur une autre dans nos futurs modèles.

Cette répartition par source s'explique par un événement créé par Kaggle autour de l'étude du COVID-19 en 2021. La **COVID-19 RADIOGRAPHY DATABASE** a été le grand gagnant du 'COVID-19 Dataset Award' de Kaggle Community.

Une équipe de chercheurs de l'université du Qatar, à Doha (Qatar), et de l'université de Dhaka (Bangladesh), ainsi que leurs collaborateurs du Pakistan et de Malaisie, en collaboration avec des médecins, ont créé une base de données d'images de radiographies thoraciques de cas positifs au COVID-19, ainsi que d'images de pneumonies normales et virales. Cet ensemble de données sur le COVID-19, les cas normaux et d'autres infections pulmonaires est publié par étapes.

Dans la première version, il a été publié 219 images COVID-19, 1341 images normales et 1345 images de radiographie pulmonaire (CXR) de pneumonie virale. Dans la première mise à jour, la classe COVID-19 a été augmentée à 1200 images CXR.

Dans la deuxième mise à jour, la base de données a été augmentée à 3616 cas positifs COVID-19 avec 10 192 images normales, 6012 images d'opacité pulmonaire (infection pulmonaire non COVID), et 1345 images de pneumonie virale et les masques pulmonaires correspondants.

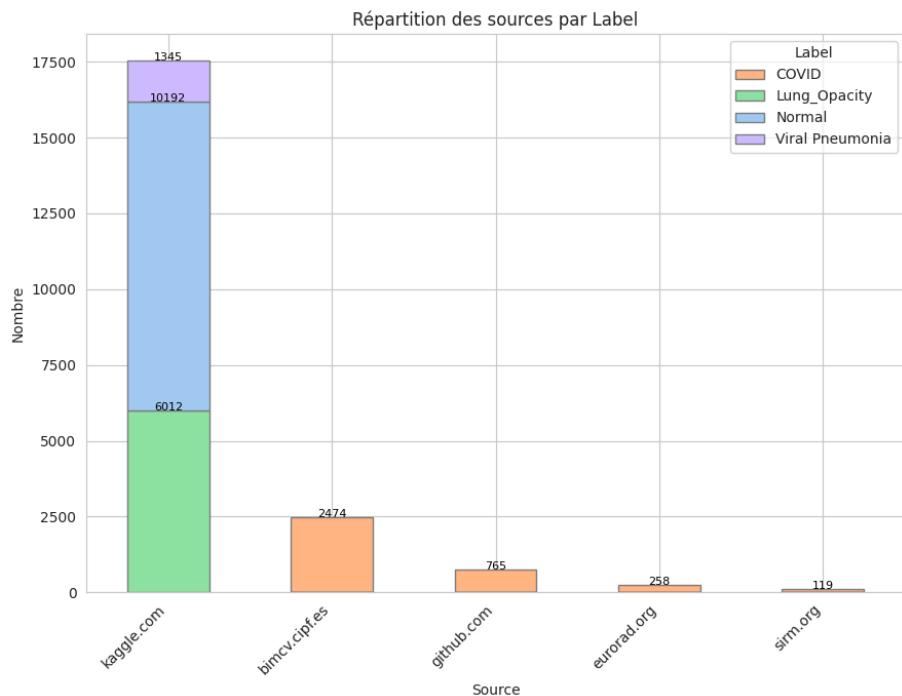


Figure 5. Proportion d'images des différents LABEL en fonction des SOURCE

Nous dénombrons en effet cinq sources différentes avec une importante part provenant de kaggle.com. Les différentes sources sont :

- **kaggle.com** : site communautaire très connu autour de la data.
- **bimcv.cipf.es** : base de données de la Medical Imaging Databank of the Valencia Region.
- **github.com** : site de partage et de contribution pour développeurs.
- **eurorad.org** : base de données médicale gérée par the European Society of Radiology (ESR).
- **sirm.org** : base de données de la Società Italiana di Radiologia Medica e Interventistica (SIRM).

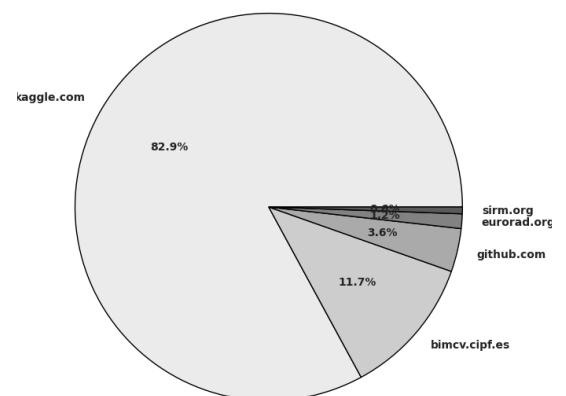


Figure 6. Proportions des différentes sources

2.3. Exploration des radiographies

Maintenant que nous avons traité les quelques données tabulaires que nous avions à notre disposition, observons quelques échantillons d'images. Nous avons extrait et affiché **cinq images aléatoires** provenant de chaque **LABEL** grâce au package **matplotlib**.

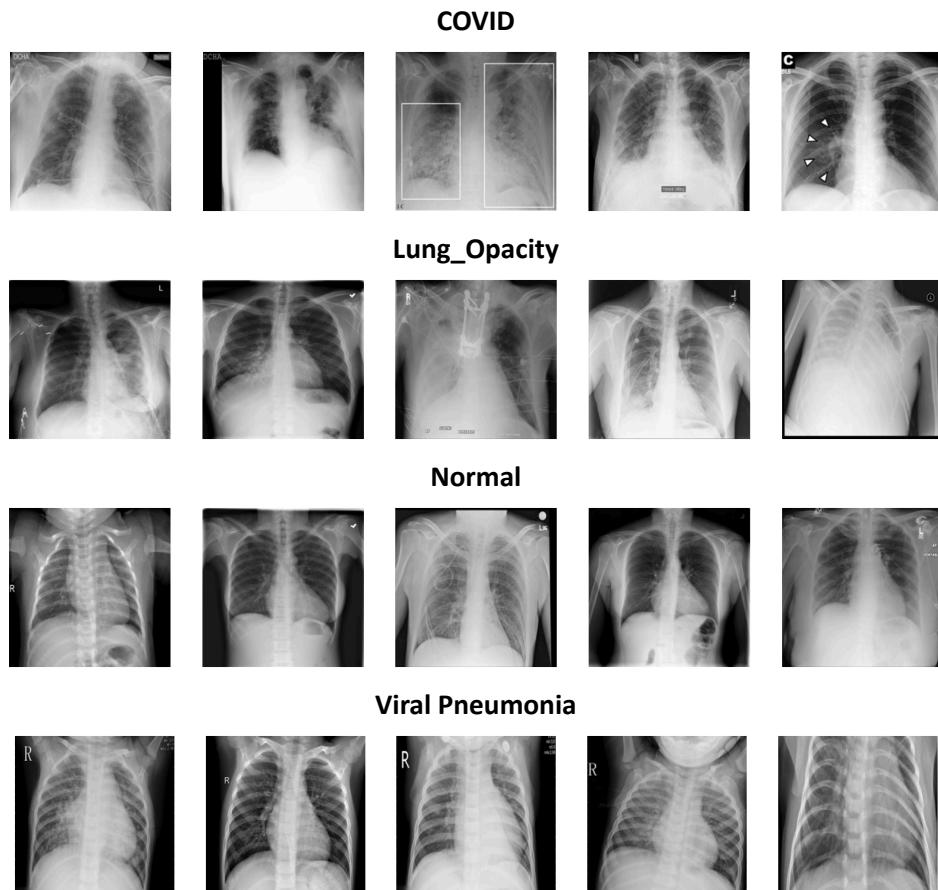


Figure 7. Échantillons aléatoires des radiographies de chaque classe

Il est remarquable de constater que toutes nos radiographies semblent bien être en nuances de gris. De plus, des logos distinctifs provenant des différents centres de radiographies ainsi que des dates sont visibles sur certaines d'entre elles. **Certaines images présentent une qualité moindre, caractérisée par une luminosité réduite, un cadrage imprécis, voire un redimensionnement artificiel** se traduisant par d'importants espaces noirs autour des images. Il est même possible d'observer **des radiographies orientées à l'envers, avec des annotations ou des erreurs de manipulation lors de la numérisation**.

Malgré ces variations, il est à noter que toutes les radiographies semblent généralement être prises sous un même angle, de face. Néanmoins, quelques variations subsistent, telles que des patients levant les bras ou des variations de zoom. Ces différences ne devraient pas poser problème, pourvu que les poumons soient bien visibles en entier sur chaque radiographie.

Il est également important de mentionner que certaines radiographies présentent **des artefacts** tels que des fils, pouvant être des sondes naso-gastriques, des stimulateurs cardiaques ou des électrodes, ce qui pourrait potentiellement affecter nos modèles.

En observant toutes ces variabilités, nous décidons de **normaliser la valeur des pixels de nos images en divisant chaque valeur de pixels par 255** (valeur maximale des canaux) afin de limiter l'amplitude entre

les valeurs. Nous en profitons pour **ne conserver qu'un seul canal** en chargeant les images en nuances de gris grâce au package `cv2` et de son argument `cv2.IMREAD_GRAYSCALE`.

A noter qu'en plus des radiographies, le jeu de données a, pour chaque image, un masque associé.

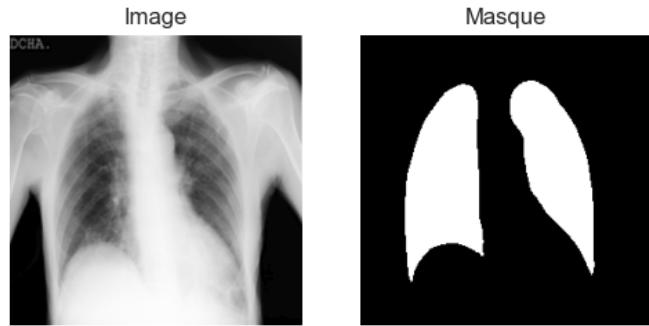


Figure 8. Image d'une radiographie et son masque basique

Il s'agit de masques basiques. Nous n'avons aucune information sur la manière dont ils ont été conçus. Nous les utiliserons cependant **comme masque de référence** sur un premier jet de modèle et des comparaisons futures. Nous envisageons également d'en créer de nouveau et de tester leur apport au travers des différentes étapes de modélisation.

2.4. Etude de la surface utile des masques

En utilisant les masques associés à chaque image, nous avons pu calculer **les ratios de la surface utile des images par classe**. On peut remarquer que pour toutes les images les informations utiles représentent 20% de la taille totale. Ceci sous-entend que **près de 80% de la surface des radiographies n'apportent pas d'informations utiles**. Ceci justifierait l'utilisation d'un masque pour que le modèle se concentre seulement sur les poumons et ne soit pas influencé par les pixels en périphérie.

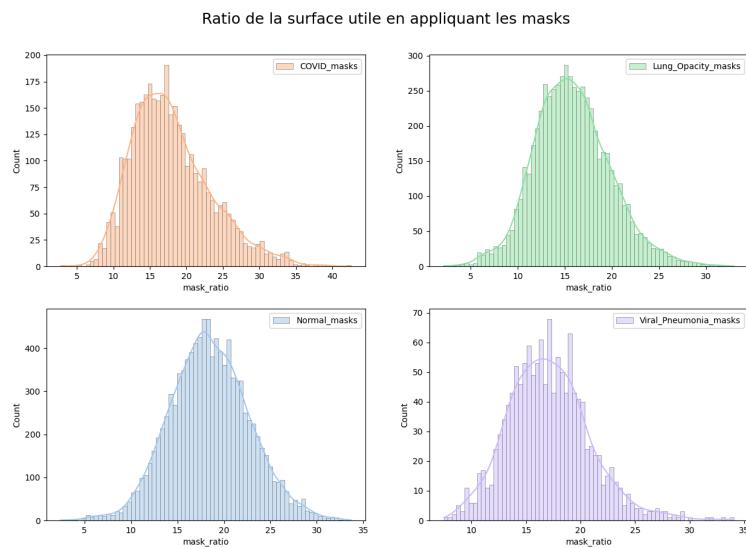


Figure 9. Distributions de la surface utile des images en appliquant les masques

2.4. Distribution de l'intensité lumineuse

Nous réalisons un violinplot afin d'étudier la distribution de l'intensité lumineuse des radiographies après normalisation et application des masques.

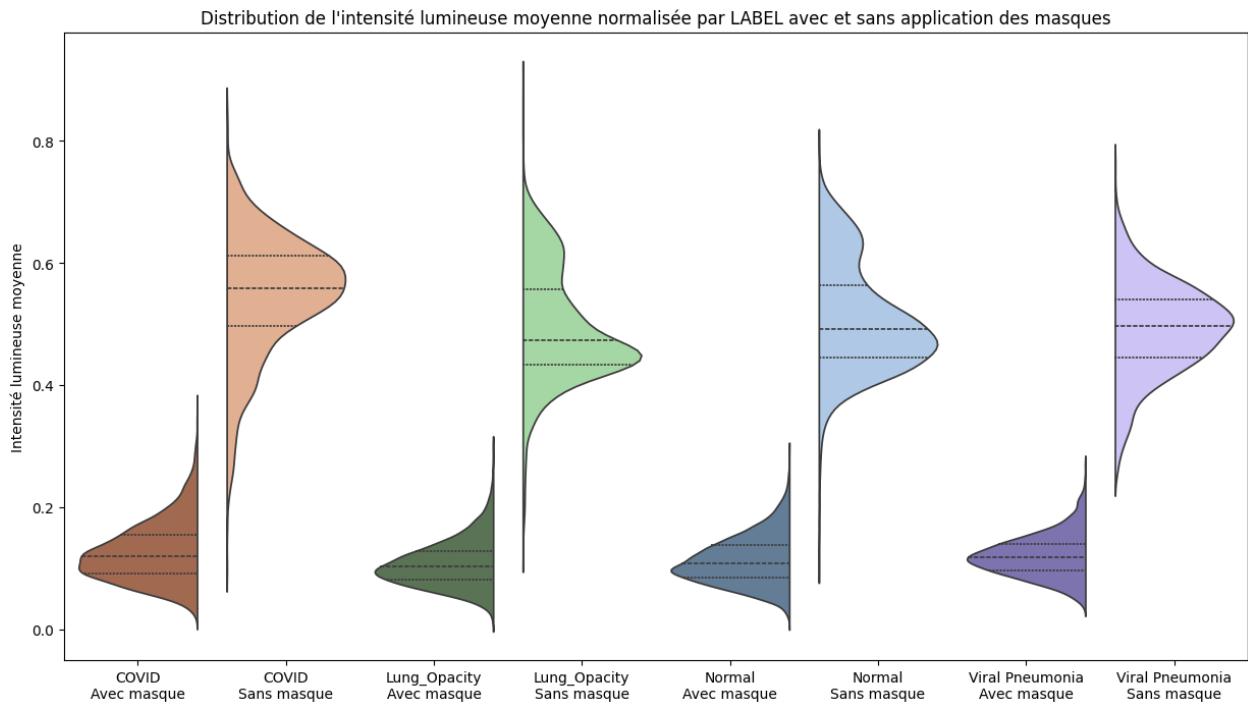


Figure 10. Distribution de l'intensité lumineuse moyenne par LABEL sans et avec les masques appliqués

Ici, la largeur des "demi-violons" donne une idée de la distribution de la probabilité de chaque classe. **Les patients de la classe COVID semblent bien avoir une distribution d'intensité plus élevée que les patients sains**, ce qui indiquerait des caractéristiques radiographiques plus prononcées dans les images des poumons.

Les classes **Lung_Opacity** et **Viral Pneumonia** présentent des distributions d'intensité élevée, ce qui est **cohérent avec la présence d'anomalies pulmonaires** qui pourraient augmenter l'intensité lumineuse moyenne dans les radiographies.

Les radiographies des patients sains ont une distribution centrée plus bas, ce qui **suggère une intensité lumineuse moins élevée**, comme nous pouvons nous y attendre.

Nous pouvons aussi constater de nombreux outliers avec des valeurs extrêmes qui, mises en relation avec nos observations précédentes, peuvent être des images de mauvaise qualité.

Nous continuons en générant des « images moyennes » pour chaque condition.

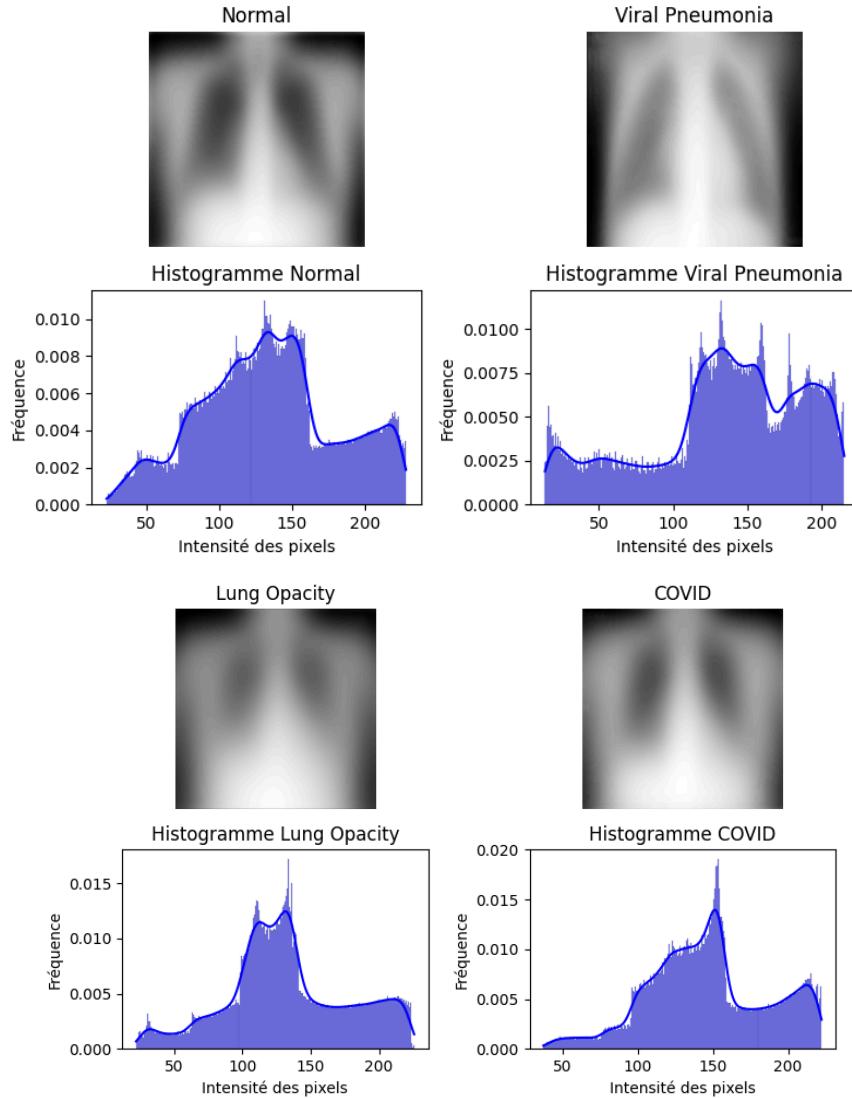


Figure 11. Fréquence de l'intensité des pixels d'une image moyenne par classe

L'image moyenne **Normal** montre des poumons noirs aux contours nets, se démarquant ainsi des autres types d'images radiographiques. En revanche, les images relatives aux conditions **Viral pneumonia** et **Lung_Opacity** révèlent des poumons plus clairs, ce qui peut indiquer une anomalie pulmonaire. L'image moyenne des radiographies de la classe **COVID**, quant à elle, se rapproche visuellement de l'aspect normal bien que les bordures pulmonaires y semblent légèrement moins définies. Cette visualisation montre aussi l'influence des éléments périphériques des poumons dans la valeur moyenne de l'intensité, **ceci explique certainement l'intérêt d'exploiter des masques pour le futur modèle.**

Pour compléter l'analyse visuelle, nous explorons la distribution de l'intensité des pixels via des histogrammes, nous procurant ainsi des données analytiques supplémentaires.

Ces histogrammes, distincts pour chaque condition, illustrent une variance notable :

- Pour **Viral Pneumonia**, l'histogramme montre une prévalence accrue de pixels à haute intensité comparativement à l'état normal, suggérant une densité différente des tissus pulmonaires.

- Dans le cas de **Lung_Opacity**, nous perdons une partie des pixels de faible intensité (valeur d'intensité entre 75 et 100) par rapport à la classe **Normal**.
- L'histogramme de la classe **COVID** présente une similarité avec celui de l'état normal mais se distingue par un pic fréquentiel autour de l'intensité 150, peut être révélateur de particularités spécifiques à cette affection virale.

3. Preprocessing des images

3.1. Traitement des images

Nous avons précédemment que bien que les radiographies soient en noir et blanc, certaines images présentent 3 canaux de couleurs.

Afin d'avoir des données uniformes, nous choisissons de les importer via OpenCV en utilisant `cv2.IMREAD_GRAYSCALE`. Nous en profitons également pour les importer en uint8 afin d'économiser de la mémoire lors des futurs traitements de nos images. **Ceci permet de convertir toutes les images en grayscale.**

Les images sont redimensionnées aux dimensions des masques, soit en 256x256 pixels. Cette étape permet de ne pas nuire à la qualité des masques par défaut, mais il est possible que nous revenions en arrière pour garder la résolution par défaut des images puis créer de nouveaux masques aux bonnes dimensions.

Les images sont donc stockées dans des array numpy. Ceci nous permet de procéder facilement à une **normalisation de l'intensité des pixels de chaque image** en les divisant par 255 (valeur maximale d'un canal) par défaut, ou alors appliquer une normalisation personnalisée si besoin en fonction des modèles que nous utiliserons par la suite.

Nous avons pu observer des différences significatives dans la répartition des différentes classes. Ceci induit qu'il va être nécessaire d'**équilibrer les classes pour éviter de donner trop de poids à certaines classes et de sous-représenter d'autres**. Nous allons essayer plusieurs possibilités, comme l'*undersampling* en s'alignant sur la classe minoritaire ou de l'*oversampling* en s'alignant sur la classe majoritaire.

Enfin, nous allons appliquer les premiers masques fournis pour **réduire la surface à seulement les informations utiles** et réduire l'influence des pixels périphériques des poumons qui pourraient influencer les modèles. Nous verrons par la suite s'il serait pertinent d'en produire de nouveaux.

Pour simplifier ces étapes, améliorer la reproductibilité et permettre les ajustements plus facilement, nous avons conçu une **fonction `preproc_img()` pour réaliser le preprocessing**. La fonction renvoie automatiquement l'ensemble d'entraînement et l'ensemble test.

```

● ● ●

def preproc_img(df_images, df_masks, n_img, normalize, files_path, resolution, with_masks):
    np.random.seed(42)

    # Gestion des erreurs
    if resolution[2] != 1 and resolution[2] != 3:
        return print("Le nombre de canaux doit être de 1 (en nuances de gris) ou de 3 (en couleur)")

    if resolution[0] != resolution[1]:
        return print("La largeur de l'image doit être la même que sa hauteur.")

    if normalize != 'imagenet' and normalize != 'simple':
        print("Attention : aucune normalisation n'a été appliquée. Utilisez 'imagenet' pour une normalisation standardisée selon le mode opératoire du set ImageNet ou 'simple' pour simplement normaliser la valeur des canaux entre 0 et 1.")

    df_images_selected_list = []
    for label, group in df_images.groupby('LABEL'):
        n_samples = min(len(group), n_img)
        df_images_selected_list.append(group.sample(n=n_samples, replace=False))
    df_images_selected = pd.concat(df_images_selected_list)

    # Initialiser une liste pour stocker les images prétraitées
    images = []

    # Sélectionner le nombre d'image à utiliser par classe
    df_masks_selected = df_masks[df_masks['FILE_NAME'].isin(df_images_selected['FILE_NAME'])] if with_masks else None

    for i in range(len(df_images_selected)):
        img_path = df_images_selected[files_path].iloc[i]
        mask_path = df_masks_selected[files_path].iloc[i] if with_masks else None

        # Charger l'image avec PIL
        img = Image.open(img_path)
        img = img.convert("L") # Convertir en niveaux de gris

        if resolution[2] == 3:
            img = img.convert("RGB") # Convertir en mode RGB

        img_resized = img.resize((resolution[0], resolution[1]))

        # Normalisation des valeurs des pixels
        if normalize == 'imagenet':
            if resolution[2] == 1: # Image en nuances de gris
                mean_gray = np.mean([0.485, 0.456, 0.406])
                std_gray = np.mean([0.229, 0.224, 0.225])
                img_normalized = (img_resized / 255.0 - mean_gray) / std_gray
            elif resolution[2] == 3: # Image en couleur
                img_normalized = np.array(img_resized) / 255.0
                img_normalized -= np.array([0.485, 0.456, 0.406])
                img_normalized /= np.array([0.229, 0.224, 0.225])
            else:
                if normalize == 'simple':
                    img_normalized = img_resized / 255
                else:
                    img_normalized = img_resized

        # Ajouter l'image à la liste
        images.append(img_normalized)

    # Reshaper pour ajouter la dimension du canal
    data = np.array(images).reshape(-1, resolution[0], resolution[1], resolution[2])
    target = df_images_selected['LABEL']

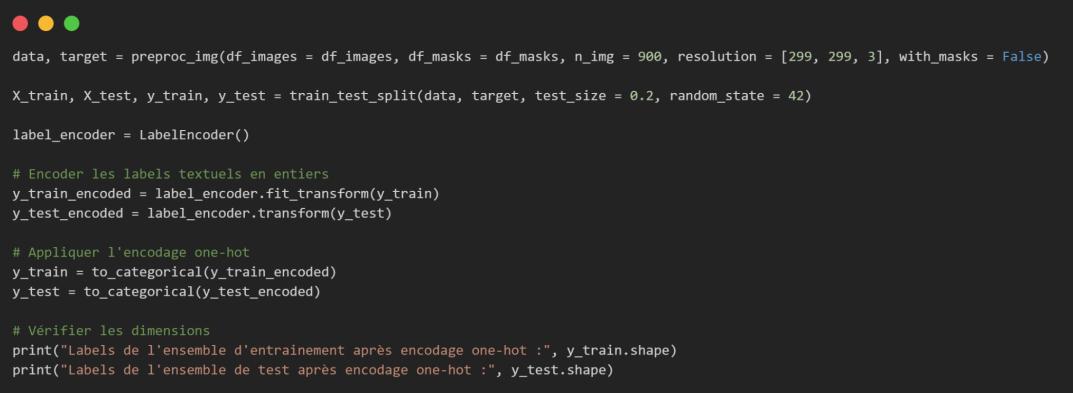
    return data, target

```

Figure 12. Code de la fonction de preprocessing

3.2. Gestion des labels

Cependant, une fois nos données traitées convenablement, **nous devons encore remplacer les labels pour que ceux-ci soient exploitables dans les modèles de classification**. Ainsi, après chaque preprocessing de nos images, nous appliquons un `LabelEncoder()` et un `to_categorical()` sur nos ensembles `y_train` et `y_test`.



```
● ● ●
data, target = preproc_img(df_images = df_images, df_masks = df_masks, n_img = 900, resolution = [299, 299, 3], with_masks = False)
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size = 0.2, random_state = 42)

label_encoder = LabelEncoder()

# Encoder les labels textuels en entiers
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Appliquer l'encodage one-hot
y_train = to_categorical(y_train_encoded)
y_test = to_categorical(y_test_encoded)

# Vérifier les dimensions
print("Labels de l'ensemble d'entraînement après encodage one-hot :", y_train.shape)
print("Labels de l'ensemble de test après encodage one-hot :", y_test.shape)
```

Figure 13. Code de la labellisation des étiquettes

Ainsi, les classes ont bien été transformées en entiers :

Classes : ['COVID' 'Lung_Opacity' 'Normal' 'Viral Pneumonia']
Numéros correspondants : [0 1 2 3]

4. Benchmarks et modélisation

4.1. Etude sur un modèle base-line simple : LeNet-5

Afin de comprendre le comportement de nos données ainsi que rechercher des ordres de grandeur pour nous guider dans la suite de la modélisation, nous sommes partis dans un premier temps sur un modèle simple, rapide à exécuter : **le modèle LeNet-5**.

Le modèle LeNet-5 est une architecture de réseau de neurones convolutionnels (CNN) largement reconnue et souvent considérée comme l'une des premières architectures dans le domaine de la *Computer Vision*. Développée par Yann LeCun et ses collègues à la fin des années 1990 puis amélioré dans les années 2000 (LeCun et al., 2016), LeNet-5 a été initialement utilisé pour la reconnaissance de caractères dans des documents dactylographiés (Darapaneni et al., 2020) mais aussi pour l'identification du genre ou des expressions faciales sur photographie (Liew et al., 2016 ; Abdullah & Abdulazeez, 2021).

Son architecture est relativement simple, elle comprend sept couches au total :

- Deux couches de convolution suivies de sous-échantillonnage (*pooling*), puis trois couches entièrement connectées et enfin une couche de sortie.
- Les deux couches de convolution sont suivies de fonctions d'activation ReLU (*Rectified Linear Unit*) et de couches de sous-échantillonnage utilisant la technique du sous-échantillonnage par moyenne.
- Les couches entièrement connectées sont suivies de fonctions d'activation ReLU, à l'exception de la couche de sortie, qui utilise généralement une fonction d'activation softmax pour produire les probabilités de sortie de chaque classe.

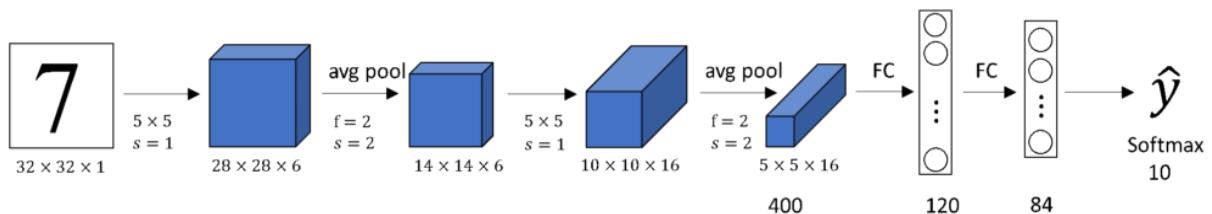


Figure 14. Schéma de l'architecture du modèle LeNet-5

Le modèle LeNet-5 prend en entrée des images en niveaux de gris de taille **32x32 pixels**. C'est une caractéristique importante étant donné que les images étaient relativement petites à l'époque où le modèle a été conçu. Les couches de convolution de taille réduite sont utilisées pour extraire des caractéristiques locales des images, suivies de sous-échantillonnage pour réduire la dimension spatiale tout en préservant les informations importantes. Ensuite, les couches entièrement connectées transforment les caractéristiques extraites en une représentation adaptée à la classification, avec des fonctions d'activation ReLU pour introduire de la non-linéarité. La couche de sortie utilise une fonction d'activation softmax pour produire des probabilités de classe.

Cette architecture a été pionnière dans la reconnaissance de caractères manuscrits mais a également jeté les bases pour de nombreuses applications de Computer vision et de Deep Learning. **De plus, cette architecture s'est déjà prêté à l'exercice de la classification de radiographies comme le décrivent les travaux de Islam et Matin (2022).**

```

● ● ●

# Initiation du modèle LeNet-5 de base
inputs = Input(shape = (28, 28, 1))

conv1 = Conv2D(filters = 30, kernel_size = (5, 5), activation = 'relu')(inputs)
pool1 = AveragePooling2D(pool_size = (2, 2))(conv1)

conv2 = Conv2D(filters = 16, kernel_size = (3, 3), activation = 'relu')(pool1)
pool2 = AveragePooling2D(pool_size = (2, 2))(conv2)

dropout = Dropout(rate = 0.2)(conv2)

flat = Flatten()(dropout)

dense1 = Dense(units = 128, activation = 'relu')(flat)
dense2 = Dense(units = 4, activation = 'softmax')(dense1)

LeNet = Model(inputs = inputs, outputs = dense2)

LeNet.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
training_history_lenet = LeNet.fit(X_train, y_train,
                                    epochs = 50,
                                    batch_size = 256,
                                    validation_split = 0.2)

```

Figure 15. Code de la structure du modèle LeNet-5

4.2. Nombre d'images optimal

Nous nous posons la question du nombre d'images nécessaire afin d'avoir des résultats optimaux, et un bon compromis entre les performances du modèles et le temps nécessaire au calcul.

Pour ce faire, nous réalisons un benchmark du nombre d'images en fonction de l'accuracy et de la validation accuracy. Ainsi, dans une boucle, **le modèle est répété avec un nombre croissant d'images par classe de 100 jusqu'à 5000 avec un pas de 100**. Nous utilisons les valeurs maximales obtenues lors des essais pour tracer notre graphique.

A noter que lorsqu'il n'y a pas assez d'images dans une classe, son maximum est conservé mais les autres classes voient leur taille augmenter, entraînant ainsi un début de déséquilibre.

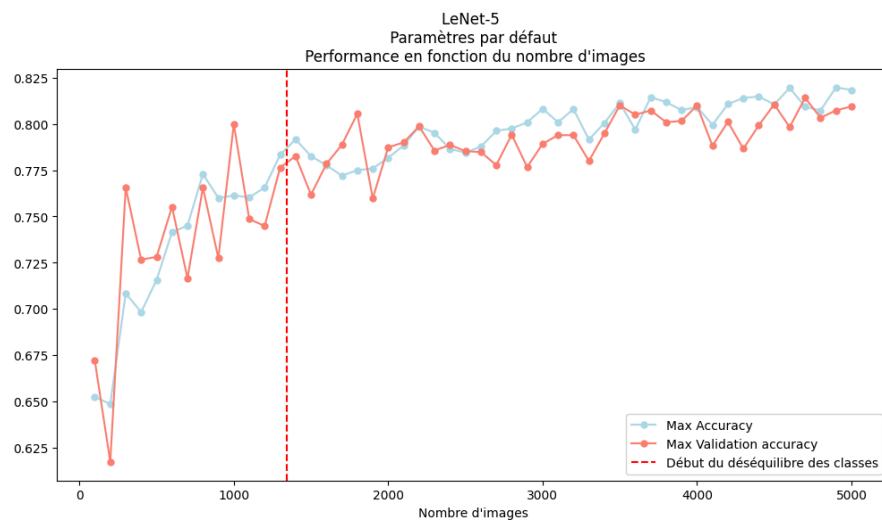


Figure 16. Courbe d'apprentissage du modèle LeNet-5 en fonction du nombre d'images utilisées

Nous pouvons observer que les performances augmentent rapidement avec le nombre d'images en début de courbe, cependant, il ne semble pas y avoir beaucoup de gain une fois le seuil de 1000 images par classe dépassé. **L'optimal semble se situer à environ 900 images par classe, au-delà duquel il ne semble pas y avoir d'évolution significative des performances.**

4.3. Nombre d'époques

De la même façon que précédemment, nous nous posons la question dans le cadre de l'optimisation du traitement de nos données et des modèles de combien d'époques il est nécessaire d'avoir recours pour avoir **un bon compromis entre les performances, temps de calcul et risque de surapprentissage**. Pour cela, nous réalisons le modèle avec 50 époques pour voir si nous observons un plateau.

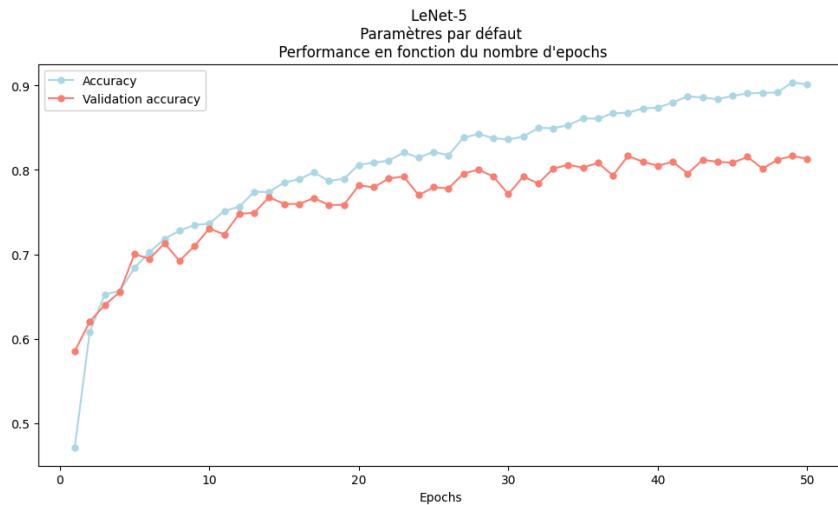


Figure 17. Courbe d'apprentissage du modèle LeNet-5 en fonction du nombre d'époques

Nous pouvons observer l'apparition d'un plateau où **l'évolution des performances ne semble plus significative à partir de 20 époques**. Nous pouvons même constater qu'au-delà de cette valeur, l'écart entre l'accuracy et la validation accuracy augmente, ce qui suggère du surapprentissage.

Nous décidons alors de conserver 20 epochs pour nos modélisations.

4.4. Usage des masques

Dans un second temps, nous appliquons le modèle avec ses paramètres par défaut sur notre jeu d'images, en appliquant les masques. Pour cela, nous devons traiter nos images, puis créer un ensemble d'entraînement et un ensemble de test. Afin de garder un équilibre entre les différentes classes tout en appliquant un maximum de masques, **nous décidons d'utiliser le maximum de la classe la moins fournie, nous prenons ainsi 1345 images par classe.**

Nous répétons exactement la même opération avec les mêmes images tirées aléatoirement sans appliquer les masques.

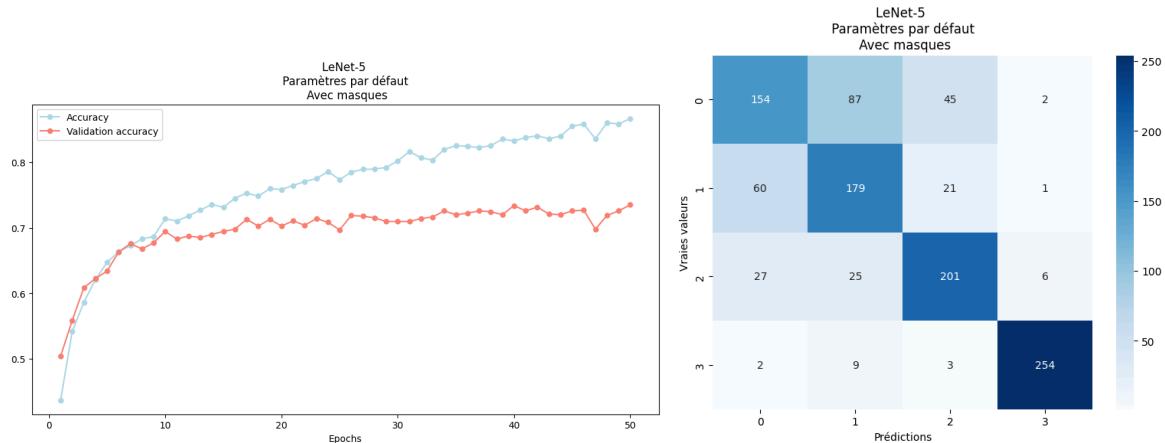


Figure 18. Courbe d'apprentissage du modèle LeNet-5 avec application des masques

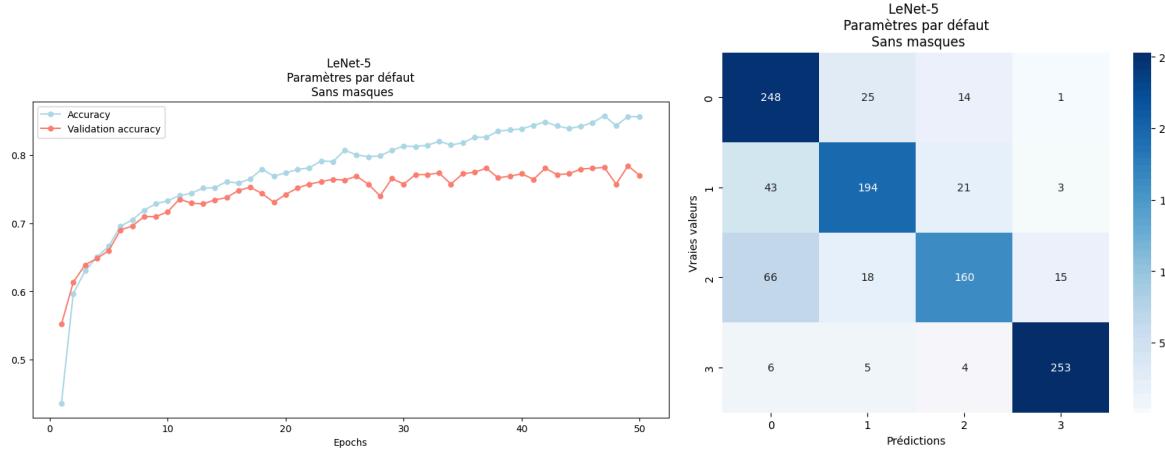


Figure 19. Courbe d'apprentissage du modèle LeNet-5 sans application des masques

Les premiers comparatifs montrent **un modèle plus précis lorsque les masques ne sont pas appliqués**. Ceci peut sûrement s'expliquer par des caractères significatifs présents en périphéries des poumons qui ne sont pas exploitées à cause de la superposition des masques.

De plus, utiliser des masques supposerait que les données d'entrée doivent toujours être appliquées d'un masque. Il serait alors nécessaire de créer un modèle adapté à la création de masque avant l'application de notre modèle de classification. Ce n'est pas l'objectif ici, **nous décidons alors de ne pas appliquer les masques par la suite**, ou d'en créer d'autres plus performants si nous en avons le temps.

5. Modèles pré-entraînés et *transfer learning*

● Paramètres communs

Tous les modèles testés dans cette partie sont des modèles complexes pré-entraînés sur le set d'images ImageNet, dont nous avons pu trouver les sources via les travaux de Kavi et al. (2020) et la documentation de Keras¹. Nous avons donc eu recours à du *transfer learning* pour classifier nos radiographies. Pour rendre les comparaisons exploitables, nous avons bien sûr fixé une base commune. **Les sources de stochasticité ont été fixées via des `np.random.seed(42)` afin de nous permettre de comparer les résultats.**

- Utilisation de 900 images aléatoires par classe et sans masque appliqué ;
- Classification du modèle sur 20 époques (si pas de *callbacks*) ;
- Optimiseur Adam avec un learning rate de 1e-4 ;
- Utilisation des paramètres par défaut des modèles avec des couches non-entraînables ;
- Ajout d'une couche dense de 256 neurones avec une fonction d'activation ReLU et d'une couche Dense de 4 neurones avec une fonction d'activation softmax pour modifier le nombre de classes de sortie.

● Normalisation

Afin d'assurer une cohérence et optimiser l'efficacité des modèles sur nos images, il est **nécessaire d'avoir recours à une normalisation similaire aux images utilisées pour l'entraînement** (du set ImageNet), hors mention contraire de certains modèles appliquant la normalisation via des couches dans son architecture.

Cette normalisation permet de rendre les images plus stables numériquement pendant l'entraînement du modèle, contribuant à une convergence plus rapide du processus d'entraînement et à des performances améliorées lors de l'inférence. Cette étape a été rajoutée à notre fonction de pré-processing précédemment décrite.

Dans un premier temps, **les images sont redimensionnées à une taille standard de 224x244 pixels** (sauf mention contraire d'une particularité d'un modèle), pour assurer une cohérence dans la taille des entrées pour le modèle. Par la suite, **les valeurs de pixel sont mises à l'échelle pour être comprises dans une plage standard de 0 à 1 en les divisant par 255** (le maximum).

Une fois les valeurs mises à l'échelle, **les données sont centrées autour de zéro pour chaque canal**, cela signifie que la moyenne de chaque canal est soustraite à toutes les valeurs de ce canal. Cette centralisation est **suivie d'une normalisation par l'écart-type de chaque canal** pour mettre à l'échelle les valeurs de manière à avoir une variance unitaire.

¹ <https://keras.io/api/applications/#usage-examples-for-image-classification-models>

- **Surveillance des métriques**

Le projet à pour objectif de classifier des radiographies selon certaines pathologies. Dans ce contexte médical, **chaque métrique a son importance et il est pertinent d'en surveiller plusieurs conjointement.**

La **précision** mesure la proportion d'observations positives correctement identifiées parmi toutes les observations identifiées comme positives par le modèle. Elle permet alors d'avoir un ratio des vrais positifs sur la somme des vrais positifs et des faux positifs. **Elle est ainsi pertinente dans notre situation pour ne pas diagnostiquer un patient sain en patient malade**, même si ce scénario est préférable à l'inverse.

Le **rappel** mesure la proportion d'observations positives correctement classées parmi toutes les observations réellement positives. Elle permet d'avoir un ratio des vrais positifs sur la somme des vrais positifs et des faux négatifs. Elle est d'autant plus importante que la précision car elle permet d'appréhender la capacité du modèle à ne pas laisser passer des observations positives, et ainsi, **à ne pas identifier un patient malade en tant que patient sain**. En effet, dans le cadre médical, une pathologie manquée peut avoir des conséquences plus graves que son inverse. De ce fait, il s'agit d'une métrique très importante à surveiller.

L'**AUC (Area Under Curve)** permet quant à elle d'évaluer les performances d'un modèle de classification en termes de sa capacité à classer correctement les observations positives et négatives. Cette métrique permet de mesurer la capacité du modèle à discriminer entre les classes en calculant l'aire sous la courbe ROC (*Receiver Operating Characteristic*). C'est une métrique particulièrement intéressante dans notre cas **où l'équilibre dans les faux positifs et vrais négatifs est important**.

Le **F1 Score** combine à la fois la précision et le rappel, permettant en une seule valeur d'évaluer la performance d'un modèle de classification. **Cette métrique nous permet d'évaluer la capacité globale du modèle à bien classer les observations en tenant compte à la fois de la précision et du rappel**. C'est ainsi une des métriques les plus importantes à surveiller dans notre contexte.

5.1. InceptionResNetV2

L'InceptionResNetV2 combine des modules Inception et des blocs de résidus pour former une architecture profonde et hautement performante. Elle se compose de plusieurs blocs d'Inception, qui permettent de capturer des informations à différentes échelles spatiales, et des blocs résiduels qui facilitent l'entraînement en profondeur en atténuant le problème de disparition des gradients (He et al., 2016).

L'architecture prend généralement une entrée des images de taille variable, **souvent 299x299 pixels avec trois canaux** dans sa configuration la plus courante.

Les modules Inception utilisent des convolutions de différentes tailles en parallèle pour capturer les caractéristiques à différentes échelles spatiales. Ces modules incluent également des couches de sous-échantillonnage pour réduire la dimensionnalité des caractéristiques. Les blocs résiduels introduisent des connexions résiduelles qui permettent un meilleur flux d'information dans le réseau et facilitent l'entraînement de réseaux profonds. Des couches de réduction de dimensionnalité sont utilisées entre les blocs d'Inception et de résidus pour réduire la dimensionnalité des caractéristiques et améliorer l'efficacité du calcul. Les activations ReLU sont utilisées après chaque opération de convolution pour introduire de la non-linéarité dans le modèle.

L'InceptionResNetV2 exploite le Deep Learning pour extraire des caractéristiques complexes à partir d'images, ce qui le rend **particulièrement adapté à des tâches telles que la classification de radiographies** comme en témoignent les travaux de Wang et al. (2019) et de An et al. (2019) sur les nodules pulmonaires.

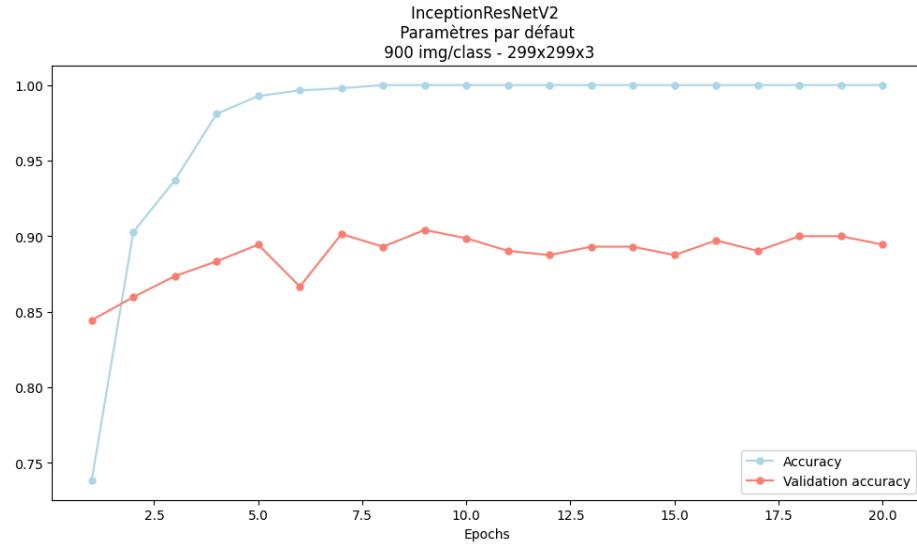


Figure 20. Courbe d'apprentissage du modèle InceptionResNetV2 avec les paramètres par défaut

L'analyse des performances du modèle InceptionResNetV2 montre des signes de surapprentissage, comme en témoignent la précision élevée sur les données d'entraînement et la stagnation (voire la diminution) de la précision de validation après les premières époques. **Cela suggère que le modèle se spécialise trop sur les données d'entraînement et pourrait ne pas généraliser aussi efficacement sur de nouvelles données.**

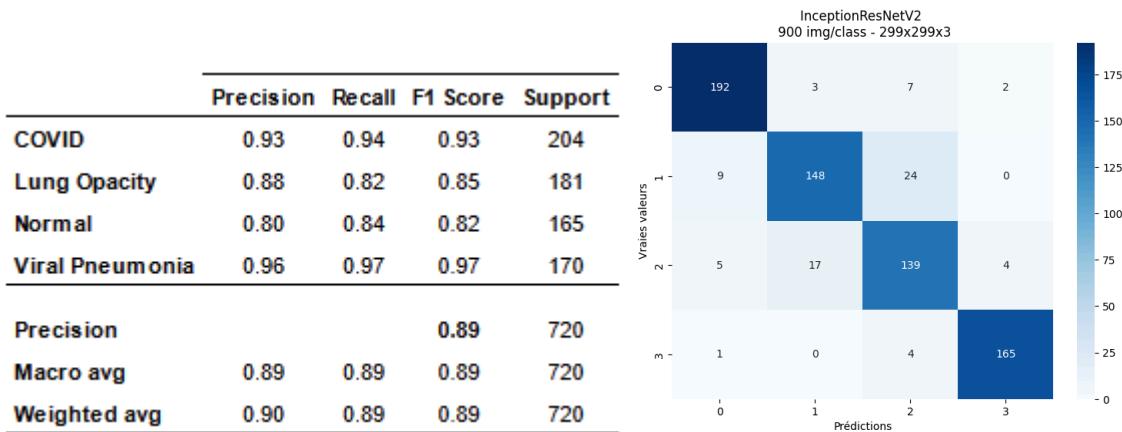


Figure 21. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle InceptionResNetV2

La matrice de confusion et les métriques détaillées par classe indiquent que le modèle a une capacité variable à distinguer les différentes classes de radiographies. La classe **Viral Pneumonia** présente d'excellents scores de précision, de rappel et de F1, indiquant une identification quasi parfaite, tandis que la classe **Normal** a montré des difficultés plus marquées, avec les scores les plus bas pour ces mêmes métriques. Le score F1, qui équilibre la précision et le rappel, suggère que le modèle est plus apte à identifier correctement les classes **COVID** et **Viral Pneumonia**, mais qu'il pourrait bénéficier d'un rééquilibrage ou d'un ajustement dans la classification des classes **Lung_Opacity** et **Normal**.

5.2. ResNet121V2

L'architecture ResNet152V2² est composée de plusieurs blocs résiduels empilés les uns sur les autres, avec une profondeur significative atteignant 152 couches dans sa configuration standard. Chaque bloc résiduel contient plusieurs couches de convolution, suivies d'une connexion résiduelle qui contourne certaines couches pour faciliter le flux d'information dans le réseau (Villain, 2021).

Cette architecture prend généralement en entrée des images de taille variable, **souvent 224x224 pixels avec trois canaux** dans sa configuration la plus courante.

Les blocs résiduels sont le composant central de l'architecture ResNet152V2. Ils introduisent des connexions résiduelles qui permettent de passer outre certaines couches de convolution, facilitant ainsi le flux d'information et aidant à résoudre le problème de la disparition des gradients dans les réseaux très profonds. Des couches de réduction de dimensionnalité sont utilisées à l'intérieur des blocs résiduels pour réduire la dimensionnalité des caractéristiques et améliorer l'efficacité du calcul. Les activations ReLU sont utilisées après chaque opération de convolution pour introduire de la non-linéarité dans le modèle. Les activations ReLU sont utilisées après chaque opération de convolution pour introduire de la non-linéarité dans le modèle. En utilisant des connexions résiduelles, ResNet152V2 favorise la stabilité de l'entraînement même pour des architectures très profondes, ce qui permet d'atteindre des performances élevées avec un entraînement plus stable et une convergence plus rapide.

ResNet152V2 est une architecture de réseau profond très performante, basée sur des blocs résiduels, qui a considérablement amélioré la capacité des réseaux neuronaux à apprendre des représentations de données complexes, tout en garantissant un entraînement stable et une convergence rapide.

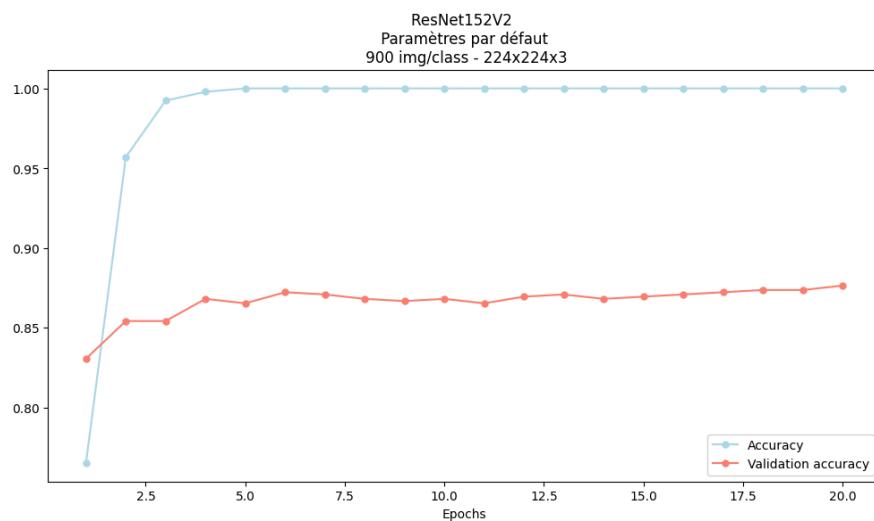


Figure 22. Courbe d'apprentissage du modèle ResNet152V2 avec les paramètres par défaut

L'analyse des performances du modèle ResNet152V2 indique des **résultats encourageants** mais aussi quelques domaines nécessitant attention pour l'analyse de radiographies dans notre contexte. La courbe d'accuracy montre une convergence stable entre les données d'entraînement et de validation, avec un écart plus faible par rapport au modèle InceptionResNetV2 précédemment analysé, **ce qui pourrait signaler une meilleure généralisation**. Cependant, l'accuracy de validation plafonne autour de 0.87, suggérant que des ajustements supplémentaires pourraient être bénéfiques.

² <https://keras.io/api/applications/resnet/>

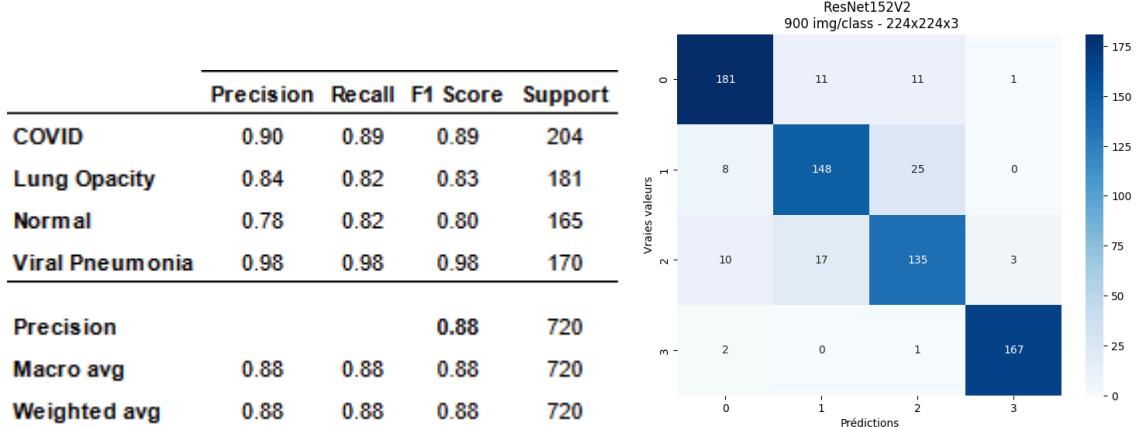


Figure 23. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle ResNet152V2

La matrice de confusion montre que le modèle a une certaine tendance à confondre la classe **COVID** avec les classes **Lung_Opacity** et **Normal**, comme en témoignent les 11 erreurs dans chaque cas. Néanmoins, la classe **Viral Pneumonia** est interprétée avec une grande précision, indiquant que les caractéristiques distinctives de cette classe sont bien capturées par le modèle. Les métriques par classe montrent que la classe 3 se distingue avec une précision et un rappel exceptionnels proches de 0.98, menant à un score F1 similaire, qui est une mesure robuste de la précision globale. Les classes **COVID**, **Lung_Opacity**, et **Normal** présentent des scores F1 légèrement plus bas, mais toujours respectables, bien que ces classes pourraient bénéficier d'un réajustement du modèle pour améliorer la distinction entre elles. **La précision globale du modèle à 0.88 est solide, mais l'objectif serait de viser une amélioration dans la classification fine entre les classes similaires.**

5.3. DenseNet201

Le modèle DenseNet201³ se compose de nombreux blocs de convolution densément connectés. Chaque couche est reliée à toutes les autres couches dans un bloc, ce qui favorise un flux direct et complet d'informations à travers le réseau.

Cette architecture prend généralement en entrée des images de taille variable, **souvent 224x224 pixels avec trois canaux** dans sa configuration la plus courante.

Chaque bloc dans DenseNet201 est composé de couches de convolution qui sont connectées à toutes les autres couches du bloc, formant ainsi une structure dense. Cette connexion dense favorise un échange d'informations riche entre les différentes couches, ce qui permet d'extraire des caractéristiques plus précises et complexes des images. Le modèle extrait des caractéristiques complexes à partir d'images, le rendant adapté à une variété de tâches de *Computer vision*, y compris pour la classification d'images comme les radiographies. En reconnectant toutes les couches à chaque bloc, DenseNet201 réduit le nombre total de paramètres par rapport à d'autres architectures similaires, ce qui améliore l'efficacité de l'apprentissage et réduit le risque de surapprentissage. Les activations ReLU sont généralement utilisées après chaque opération de convolution pour introduire de la non-linéarité dans le modèle. La structure dense favorise alors la stabilité de l'entraînement en encourageant le flux direct d'informations à travers le réseau, ce qui facilite la convergence et améliore la robustesse du modèle.

DenseNet201 est une architecture de réseau neuronal profond qui se caractérise par sa connexion dense entre les couches, ce qui favorise un échange optimal d'informations. **Il s'agit d'une structure**

³ <https://keras.io/api/applications/densenet/>

particulièrement intéressante pour analyser finement les différentes features des radiographies (Aziz et al., 2021 ; Jaiswal et al., 2021). Cette architecture est efficace pour extraire des caractéristiques complexes à partir d'images, avec une utilisation efficace des paramètres et une stabilité d'entraînement.

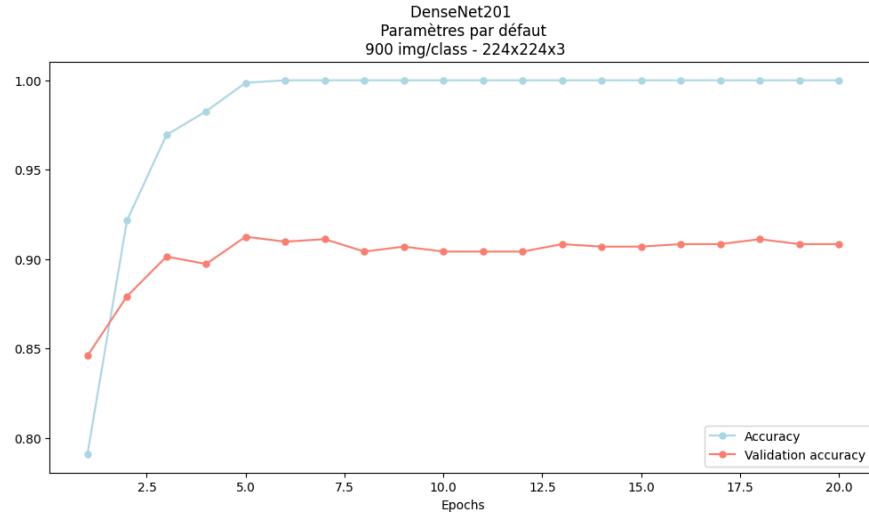


Figure 24. Courbe d'apprentissage du modèle DenseNet201 avec les paramètres par défaut

Le DenseNet201 réalise une classification robuste sur les radiographies. La courbe d'accuracy illustre une précision élevée et stable pour les données d'entraînement, avec une précision de validation légèrement inférieure mais bien maintenue autour de 0.90 au-delà de la cinquième époque. **Ceci suggère une bonne généralisation du modèle sans signes évidents de surapprentissage.** La légère baisse de la précision de validation par rapport à l'accuracy d'entraînement pourrait indiquer que le modèle peut encore bénéficier d'une optimisation fine pour rapprocher ces deux mesures.

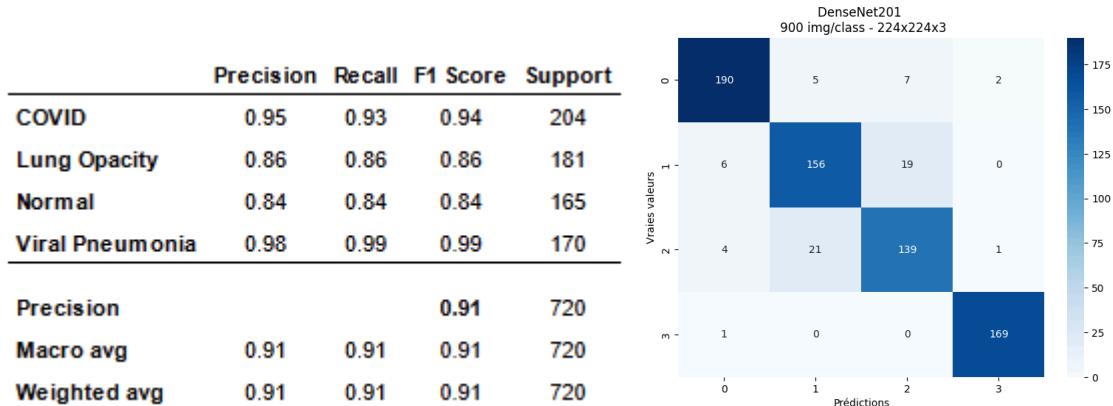


Figure 25. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle DenseNet201

La matrice de confusion confirme l'efficacité du modèle avec une majorité d'images correctement classées dans leurs catégories respectives. Les erreurs de classification les plus courantes semblent se produire entre les classes **Lung_Opacity** et **Normal**, sous-entendant des similarités entre les caractéristiques des radiographies que le modèle confond certainement. Selon le tableau de métriques, le modèle a une excellente précision pour la classe **COVID** et des scores exceptionnels de rappel et de F1 pour la classe **Viral Pneumonia**, indiquant une classification presque parfaite pour ces catégories. Les classes **Lung_Opacity** et **Normal** ont des scores F1 légèrement inférieurs mais comparables. **Tout ceci indique une bonne performance de classification qui reste uniforme entre ces catégories.**

5.4. VGG16

Le modèle VGG16⁴ est un réseau de neurones convolutifs profond développé par l'université d'Oxford en 2014. Il a été l'un des modèles les plus performants lors de la compétition ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*) la même année.

Ce modèle est composé de 16 couches profondes comprenant : 13 couches de convolutions, 5 couches de pooling et 3 couches entièrement connectées. Les filtres de convolution de taille 3x3, avec un pas de 1, permettent une capture efficace des informations spatiales, tandis que les couches de pooling, utilisant un max pooling 2x2 pixels et un pas de 2, réduisent la taille des cartes d'activation tout en préservant les informations essentielles. Les 3 couches finales ont pour rôle la classification dans 1000 classes différentes (catégories ImageNet).

Avec ses 16 couches, le modèle VGG16 est considéré comme un modèle dit "profond" et performant malgré son architecture simple (répétition de blocs convolution-pooling). En effet, VGG16 a obtenu d'excellents résultats sur la tâche de classifications d'imageNet (erreur top-5 de 7%).

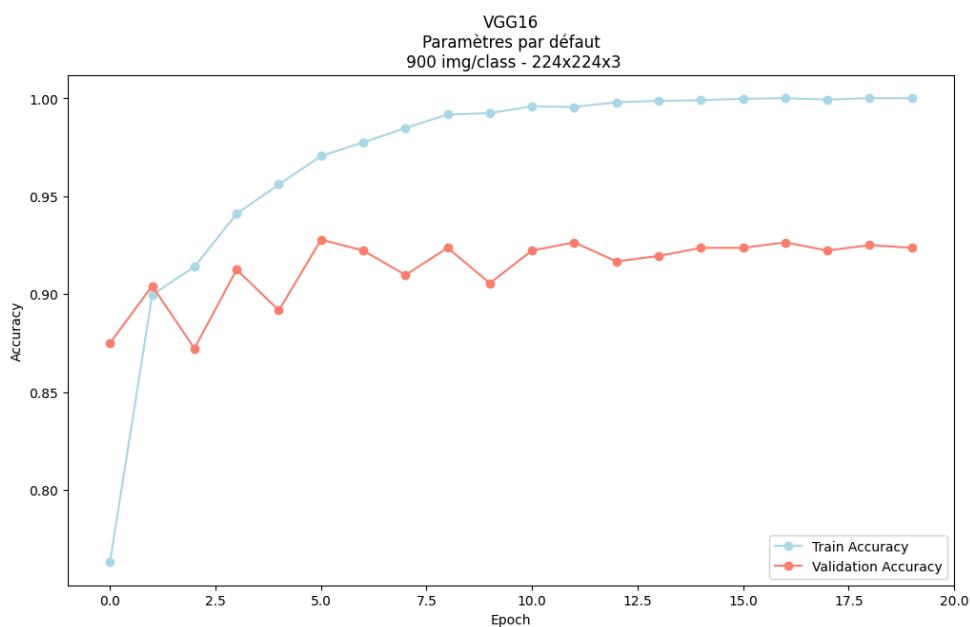


Figure 26. Courbe d'apprentissage du modèle VGG16 avec les paramètres par défaut

VGG16 obtient de bonnes performances sur la classification des différentes radiographies. La précision de l'entraînement augmente de façon significative lors des premières époques avant de se stabiliser autour d'une valeur proche de 1. La précision obtenue pour l'ensemble d'entraînement est élevée dès le départ et se stabilise à partir de 10 époques aux alentours de 0.92. **L'écart entre les valeurs auxquelles se stabilisent les précision d'entraînement et de validation suggère qu'il doit être possible d'améliorer encore les résultats en faisant bénéficier le modèle d'ajustements supplémentaires.**

⁴ <https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide>

	Precision	Recall	F1 Score	Support
COVID	0.97	0.92	0.95	192
Lung Opacity	0.87	0.9	0.88	173
Normal	0.85	0.88	0.86	176
Viral Pneumonia	0.98	0.98	0.98	179
Precision			0.92	720
Macro avg	0.92	0.92	0.92	720
Weighted avg	0.92	0.92	0.92	720

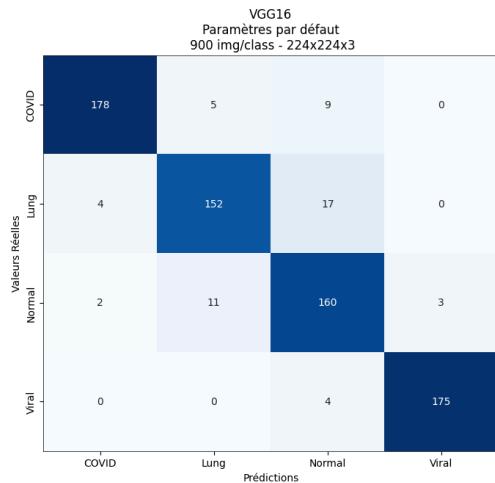


Figure 27. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle VGG16

Le rapport de classification et la matrice de confusion montrent que le modèle parvient à très bien classer les radiographies des classes **Viral** (**Viral pneumonia**) et **COVID**. Également, même si les résultats restent bons, le modèle commet plus d'erreurs de classification entre les catégories **Normal** et **Lung** (**Lung_opacity**). **Sans ajustement particulier, ce modèle semble déjà prometteur quant à ses capacités à classifier nos radiographies correctement.**

5.5. VGG19

A l'instar du modèle VGG16, le modèle VGG19 a également été conçu par l'université d'Oxford en 2014. C'est une version plus profonde de VGG16 avec 19 couches, au lieu de 16, organisées de la manière suivante : 16 couche de convolutions utilisant des filtres 3x3 et un pas de 1, 5 couches de max-pooling 2x2 avec un pas de 2, 3 couches finales fully connected.

Les couches de convolution sont regroupées en 5 blocs, chacun suivi d'une couche de max-pooling. La profondeur du réseau permet d'extraire des caractéristiques de plus en plus complexes au fur et à mesure de la propagation dans le réseau.

Comme VGG16, il a obtenu de très bon résultats sur le défi ImageNet, avec un top-5 error rate de 7.5% et possède un bon potentiel en classification d'image comme l'illustrent les travaux de Lagunas et Garces (2018)

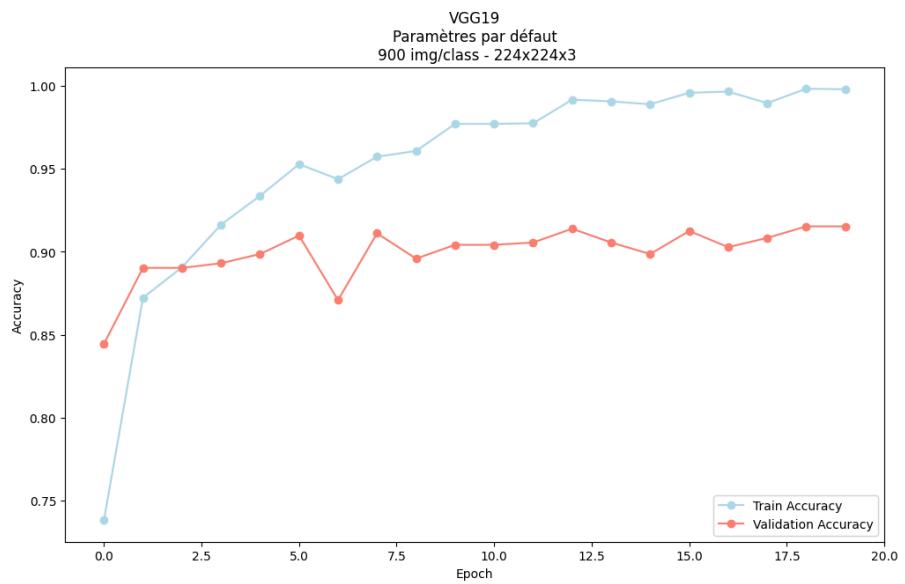


Figure 28. Courbe d'apprentissage du modèle VGG19 avec les paramètres par défaut

La précision, sur les données d'entraînement, augmente régulièrement pour atteindre pratiquement 1 à l'époque 19. Sur les données de validations la précision démarre également assez haut (0.84) pour s'améliorer assez peu au cours de l'entraînement avant de se stabiliser autour de 0.92.

	Precision	Recall	F1 Score	Support
COVID	0.92	0.95	0.93	192
Lung Opacity	0.84	0.91	0.88	173
Normal	0.91	0.81	0.86	176
Viral Pneumonia	0.97	0.97	0.97	179
Precision			0.91	720
Macro avg	0.91	0.91	0.91	720
Weighted avg	0.91	0.91	0.91	720

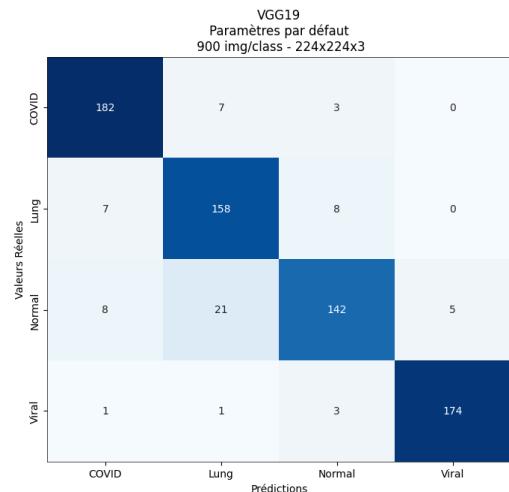


Figure 29. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle VGG19

Au vue de la matrice de confusion et du rapport de classification, les résultats obtenus semblent également très bons et superposables à ceux que nous avons obtenus pour que pour VGG16. **Cependant ce modèle étant un peu plus profond, il demande des ressources computationnelles plus importantes sans que cela ne se répercute de façon évidente sur ses performances.**

5.6. ConvNextTiny

ConvNextTiny⁵ est un modèle de réseau de neurones convolutif développé par les chercheurs de Meta AI. Il fait partie de la famille des modèles ConvNeXt, qui visent à améliorer les performances des CNN tout en réduisant la complexité du modèle.

ConvNeXtTiny est composé de quatre stades de blocs ConvNeXt, chacun contenant plusieurs couches de convolution, de normalisation et d'activation. Cette architecture permet d'extraire des caractéristiques visuelles à différentes échelle spatiales. Les blocs ConvNeXt sont caractérisés par l'utilisation de convolutions dépliées pour réduire le nombre de paramètres, l'ajout de connexions résiduelles et pour faciliter l'entraînement et l'utilisation de la normalisation des couches.

ConvNeXtTiny utilise également des couches de convolution avec un pas de 2 pour réduire la résolution spatiales plutôt que des couches de pooling.

Sur le jeu de données ImageNet, ConvNeXtTiny obtient de bonnes performances malgré son nombre de paramètres réduit (28 millions).

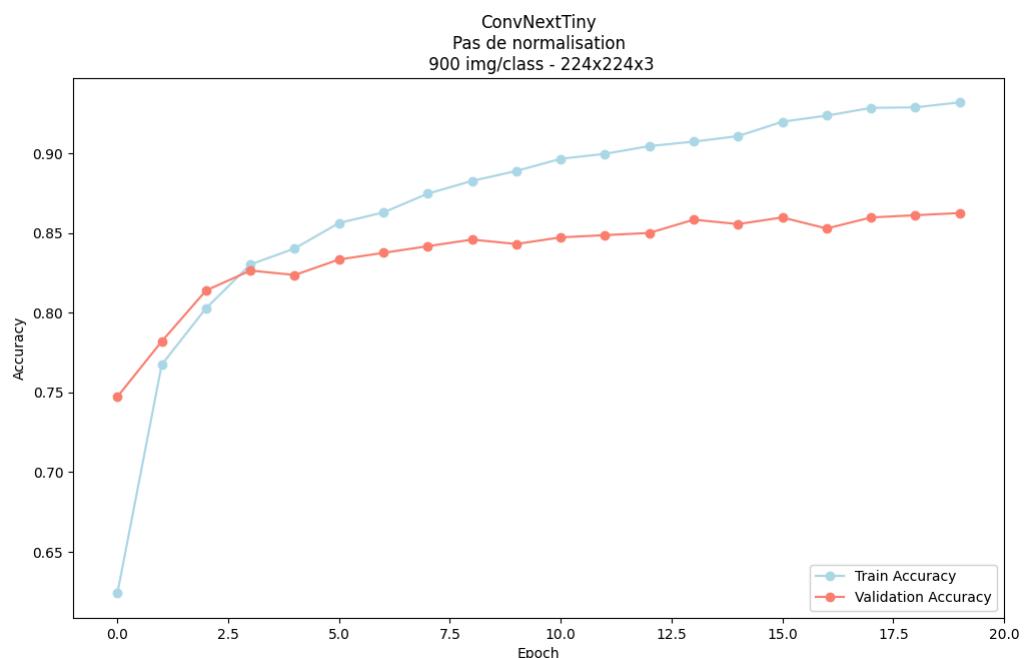


Figure 30. Courbe d'apprentissage du modèle ConvNextTiny avec les paramètres par défaut et sans normalisation

Sur les données d'entraînement la précision progresse régulièrement pour arriver à 0.93 en 20 époques. De la même manière, sur les données de validation, la précision augmente également de façon lente et régulière mais sans dépasser les 0.86 en fin d'entraînement.

⁵ <https://keras.io/api/applications/convnext/>

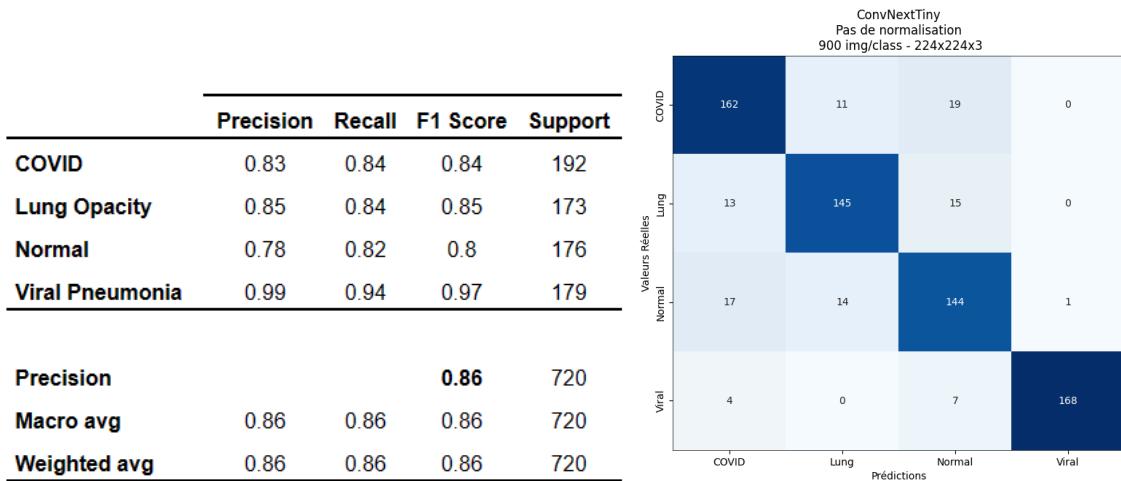


Figure 31. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle ConvNextTiny

Avec ce modèle il apparaît que la classification est significativement meilleure pour la catégorie **Viral** (**Viral pneumonia**) que pour les autres. Ceci donne un score global en deçà de ce que nous avons pu observer sur d'autres modèles dans les mêmes conditions de test. Les courbes d'apprentissage suggèrent que le modèle pourrait bénéficier d'un nombre d'époques supérieur pour continuer à s'améliorer.

5.7. ConvNextBase

ConvNeXtBase⁶ est le modèle phare de la famille ConvNeXt ,ConvNeXtTiny en était une version plus légère et rapide, et a donc également été développé par les équipes de Meta AI. Ce modèle s'appuie sur une architecture de type transformers, combinée à des couches de convolution traditionnelles. ConvNeXtBase comporte 50 couches profondes utilisant à la fois la capacité de capture des motifs locaux des convolutions et la capacité de modélisation des dépendances à longue distance des transformers.

ConvNeXtBase est connu pour égaler, voire surpasser, des modèles comme ResNet-50 ou ViT-Base et pour se montrer robuste face aux perturbations comme le bruit, le flou ou les occultations.

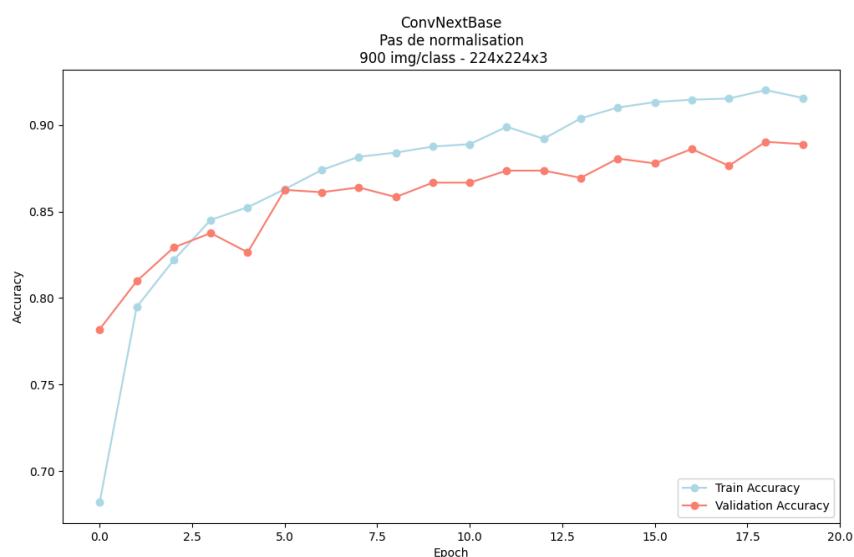


Figure 32. Courbe d'apprentissage du modèle ConvNextBase avec les paramètres par défaut et sans normalisation

⁶ <https://keras.io/api/applications/convnext/>

Comparativement à ConvNeXtTiny, ConvNextBase obtient des résultats supérieurs dans les conditions de notre protocole de test. Avec les données d'apprentissage, la précision atteint plus de 0.92 en 20 époques. Sur les données de validation l'entraînement se termine sur une valeur de 0.89.

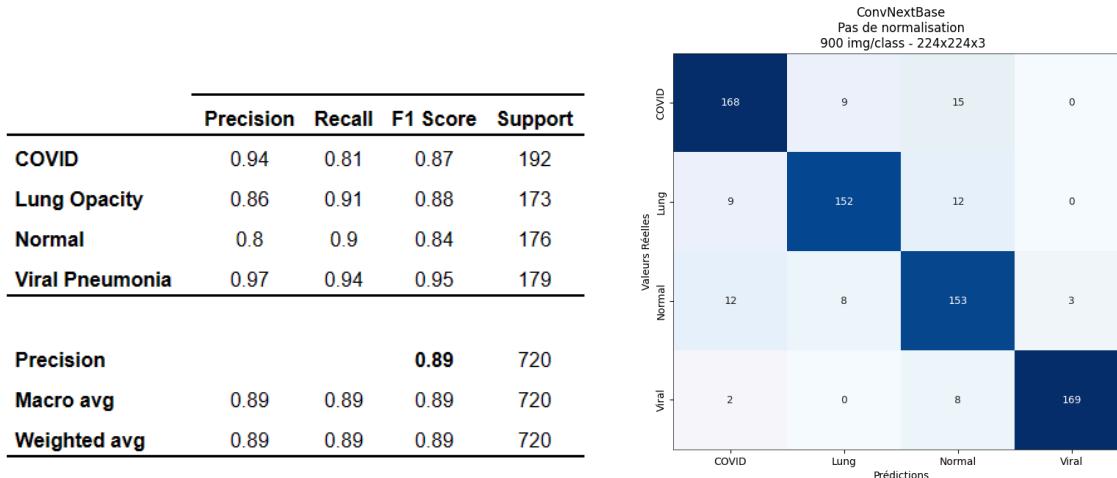


Figure 33. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle ConvNeXtBase

La classe (**Viral pneumonia**) reste toujours la mieux détectée, suivie de la classe **COVID**. Les résultats obtenus ici sont donc comparables à ceux obtenus avec le modèle ConvNeXtTiny. Encore une fois le modèle semble pouvoir bénéficier d'un allongement de la durée d'entraînement. Cependant il est à noter que ce modèle peut se montrer gourmand en termes de ressource computationnelle, **une époque de ConvNeXtBase pouvant prendre entre deux et trois fois plus de temps que le modèle ConvNeXtTiny sans montrer une différence flagrante de performance.**

6. Finetuning des modèles sélectionnés

6.1. Cas des EfficientNet

Dans l'article publié par Tan & Le (2019), les modèles EfficientNet⁷ ont été introduits. Le travail a été centré sur l'architecture du modèle en équilibrant la profondeur, la largeur et la résolution du réseau. Afin d'améliorer les performances du modèle, l'optimisation des hyperparamètres est indispensable, ce qu'on appelle Le *scaling* ou la mise à l'échelle. Une nouvelle méthode a été proposée, *compound method*, à l'issue de cette étude, qui met uniformément à l'échelle toutes les dimensions (profondeur, largeur et résolution) à l'aide d'un coefficient composé, voir la figure ci-dessous.

⁷ <https://keras.io/api/applications/efficientnet/>

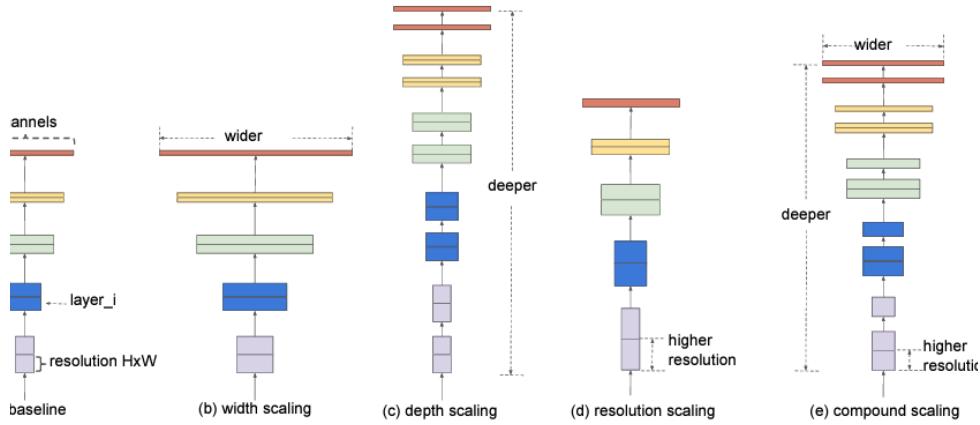


Figure 34. Architecture de modèles (*Model Scaling*), (e) compound scaling pour les modèles EfficientNet

Cette méthode a été appliquée sur les modèles MobileNets et ResNet. Une famille de modèles appelés EfficientNets a été conçue à partir d'un réseau de base, puis mis à l'échelle avec la *compound method*. **Les modèles atteignent une précision et une efficacité bien meilleures que les ConvNets précédents.** En particulier, notre EfficientNet-B7 atteint une précision de pointe de 84,3 % sur ImageNet, tout en étant 8,4 fois plus petit et 6,1 fois plus rapide pour l'inférence que le meilleur ConvNet existant. Ces modèles se transfèrent également bien et atteignent une précision de pointe sur CIFAR-100 (91,7 %), Flowers (98,8 %) et 3 autres ensembles de données d'apprentissage par transfert, avec un ordre de grandeur de moins de paramètres. Comme le montre la figure ci-dessous.

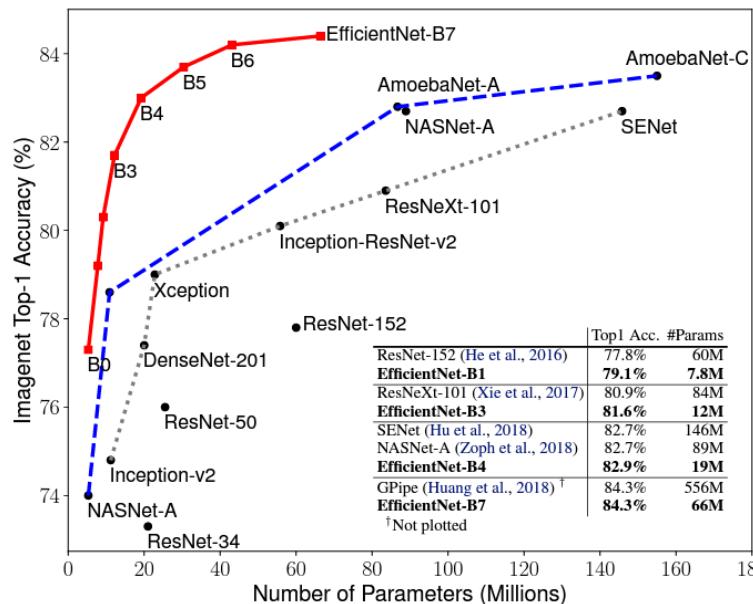


Figure 35. Taille du modèle vs. la précisions avec la base de données ImageNet

Le choix de la résolution, de la profondeur et de la largeur est limité par de nombreux facteurs :

- **Résolution** : Les résolutions qui ne sont pas divisibles par 8, 16, etc. entraînent un calage zéro près des limites de certaines couches, ce qui gaspille les ressources informatiques. Cela s'applique particulièrement aux petites variantes du modèle, c'est pourquoi la résolution d'entrée pour B0 et B1 est de 224 et 240.

- **Profondeur et largeur** : les blocs de construction d'EfficientNet exigent que la taille des canaux soit un multiple de 8.
- **Limitation des ressources** : la limitation de la mémoire peut entraver la résolution lorsque la profondeur et la largeur peuvent encore augmenter. Dans une telle situation, l'augmentation de la profondeur et/ou de la largeur tout en conservant la résolution peut encore améliorer les performances.

Par conséquent, la profondeur, la largeur et la résolution de chaque variante des modèles EfficientNet sont choisies empiriquement et ont prouvé qu'elles produisaient de bons résultats, bien qu'elles puissent s'écarte de manière significative de la formule de mise à l'échelle composée (*compound method*). (Tan & Le, 2019)

6.1.1 EfficientNets benchmark

Afin de savoir quel est le modèle le plus adapté pour notre jeu de données, un *benchmark* de des modèles EfficientNet a été fait. **Pour des raisons de limites de capacités matériel et la lourdeur des calculs, seules les versions allant de B0 à B6 ont été testées, à l'exception du EfficientNet-B7.**

Quelques spécifications :

- Les modèles EfficientNet prennent des résolutions décroissantes en fonction de la version choisie, allant de 224 x 224 pour le B0 jusqu'à 600 x 600 pour le B7.
- Les intensités de pixels des images doivent rester dans l'intervalle [0 ; 255]

Le *benchmark* a été réalisé avec les images radio de base **sans masque ni preprocessing, 900 images par classe**. L'ensemble de test a été formé à partir de 20% des données du jeu sélectionné. Avec une construction fonctionnelle d'un modèle *transfer learning*, une couche `Flatten()` est appliquée à la sortie du modèle de base (`base_model.output`), où `base_model` est le modèle de base EfficientNet B0-B6. Ensuite une couche `Dense()` avec 256 neurones, suivie d'une couche de danse pour adapter le modèle à notre problème de classification de quatre classes pour les prédictions. L'entrée du modèle fonctionnel est `base_model.input`, et la couche prédition (couche `Dense()` avec quatre neurones) en sortie.

Les modèles de base ont été **chargés avec les poids pré-entraînés sur la base de données ImageNet**. En excluant la couche `Dense()` finale (`include_top = False`) qui transforme les 1280 caractéristiques de l'avant-dernière couche en prédition des 1000 classes ImageNet. Finalement, **toutes les couches du modèle de base restent gelées** dans ce cas.

Le modèle est compilé avec un taux d'apprentissage fixe à 1e-4, la fonction de perte utilisée ici est la `categorical_crossentropy` qui est adaptée aux problèmes de classification multiple, et la métrique accuracy. L'entraînement a été effectué sur 20 époques, en gardant la taille du lot par défaut. L'ensemble de validation est constitué de 20% de l'ensemble d'entraînement.

EfficientNet models benchmark, 900 images, 20 epochs

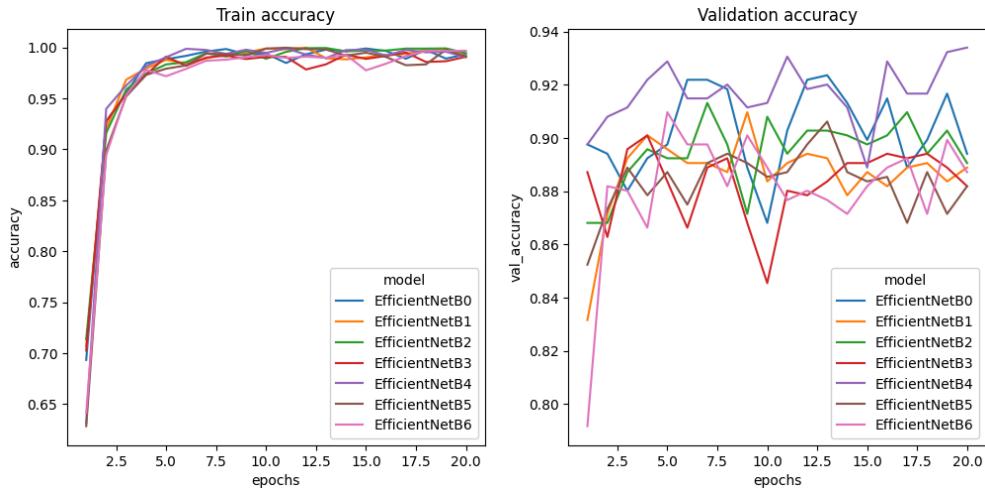


Figure 36. Benchmark transfer learning des modèles EfficientNet

Les résultats sont très intéressants, les modèles EfficientNet montrent leurs efficacité, avec un nombre de paramètres très faible comparé à d'autres modèles, en atteignant des précisions assez élevées dès les premiers epochs. Notamment, le modèle B0 qui atteint 0.92 de précision au bout de six epochs et B4 arrive à 0.93 avec seulement cinq epochs, comme le montre la figure ci-dessus. **D'où le choix de continuer avec le modèle B4 pour la suite de l'étude.** Le modèle B0 aurait été un très bon choix également, qui atteint des performances notables tout en étant moins complexes.

6.1.2 EfficientNet B4

Il était intéressant de tester les performances du modèle de base EfficientNet B4, en gardant le même jeu de données, mais seulement avec une couche `Dense()` pour faire la classification. L'entraînement s'est effectué avec les mêmes paramètres.

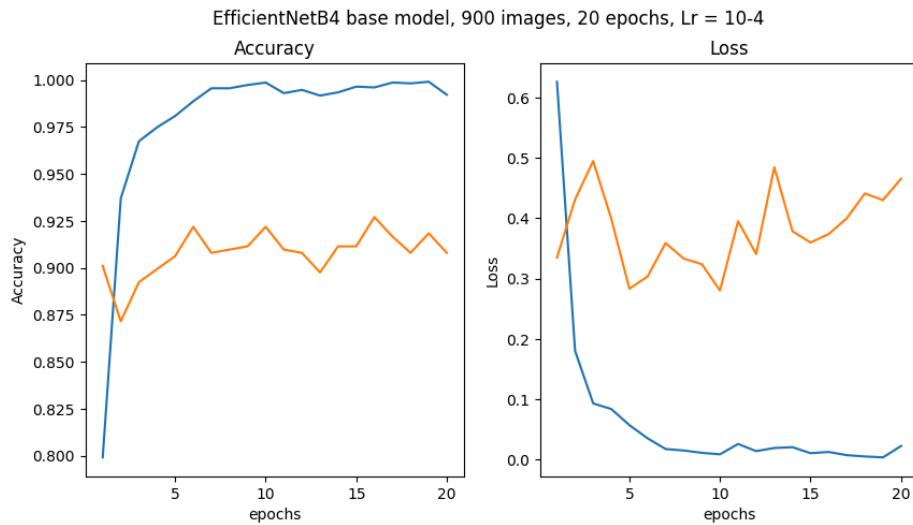


Figure 37. Courbe d'apprentissage du modèle EfficientNetB4 avec les paramètres par défaut

Le modèle est assez performant, il atteint une précision de 0.925 au bout de six époques durant l'entraînement. **Néanmoins, la perte pour l'ensemble de validation augmente ce qui indique un surapprentissage du modèle.** Cet *overfitting* peut être à cause de plusieurs raisons : le jeu de données n'est pas assez suffisant par rapport à la complexité du modèle, ou il faut un plus d'ajustement notamment sur le taux de *dropout*.

	Precision	Recall	F1 Score	Support
COVID	0.97	0.87	0.91	204
Lung Opacity	0.83	0.82	0.82	181
Normal	0.76	0.92	0.83	165
Viral Pneumonia	1.00	0.93	0.96	170
Precision			0.88	720
Macro avg	0.89	0.88	0.88	720
Weighted avg	0.89	0.88	0.88	720

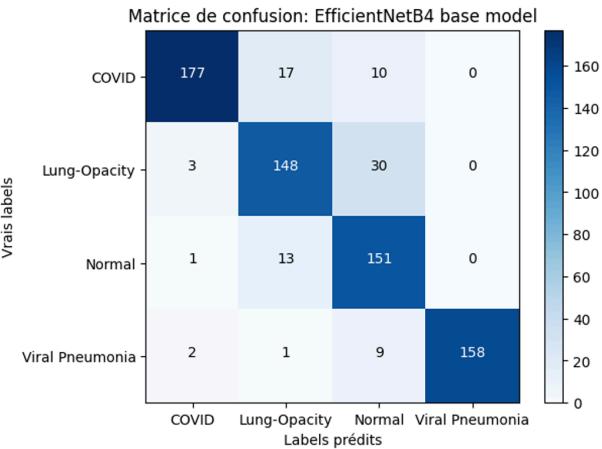


Figure 38. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle EfficientNetB4

La précision du modèle chute à 0.88 sur l'ensemble test. La détection de la classe **COVID** n'est pas au niveau de ce que l'on espérait avec une précision de 0.91. **Globalement, le modèle avec ce paramétrage donne de bons résultats.** Dans la section suivante, nous allons essayer un ajustement plus fin pour avoir de meilleures performances avec ce modèle.

6.1.3 Keras Tuner pour l'EffcientNet B4

Keras Tuner est un module qui permet de réaliser une étude d'optimisation des hyperparamètres afin de trouver les meilleures combinaisons de paramètres, permettant **d'ajuster un peu plus finement le modèle** (O'Malley et al., 2019).

Dans cette étude, les couches du modèle de base restent gelées (poids du pré-entraînement sur le jeu de données ImageNet). Afin d'intégrer les couches de classification une couche **Flatten()** est appliquée à la sortie du modèle de base. Ensuite, deux couches **Dense()** ont été utilisées avec un taux de *dropout* de 20%, et une couche **Dense()** finale pour les prédictions de nos quatre classes.

Keras Tuner permet de réaliser des études paramétriques en combinant tous les hyperparamètres spécifiés dans le modèles, en l'occurrence : le nombre de neurones des deux couches **Dense()**, et le taux d'apprentissage (*Learning Rate*), tel que : **le nombre de neurones est dans l'intervalle [32 ; 1024] avec un pas de 32, et le taux d'apprentissage est choisis parmi les trois valeurs suivantes [1e-4 ; 1e-3 ; 1e-4].** La compilation des modèles se fait avec les mêmes paramètres à savoir la fonction de perte **categorical_crossentropy** et la métrique de précision.

Il existe plusieurs fonctions intéressantes pour la recherche de paramètres optimaux pour un ajustement plus fin des modèles. **RandomSearch()** est très pratique pour chercher de manière aléatoire ces hyperparamètres optimaux, elle prend en argument le modèle, la métrique à maximiser, les paramètres à faire varier, etc. La fonction génère des combinaisons aléatoires d'hyperparamètres, et construit des modèles avec ces derniers. Puis, l'entraînement s'effectue sur l'ensemble de données d'entraînement, chaque modèle est évalué. Une fois que toutes les combinaisons d'hyperparamètres ont été évaluées, la

fonction `RandomSearch()` renvoie le modèle ayant obtenu les meilleures performances selon la métrique d'évaluation spécifiée.

La meilleure combinaison trouvée pour le modèle testé ici est: **736 neurones pour la première couche dense, 352 pour la deuxième et un taux d'apprentissage à 1e-4** (cette valeur semble être un meilleur compromis performance/coût de calcul pour plusieurs modèles).

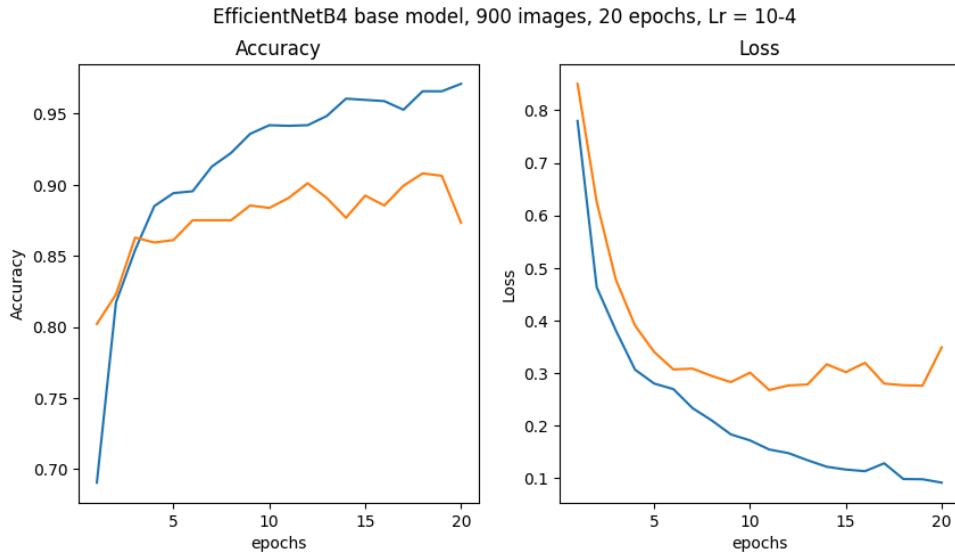


Figure 39. Performance d'entraînement du meilleur modèle issu de Keras Tuner RandomSearch, modèle de base EfficientNetB4

La figure ci-dessus présente les performances d'entraînement du modèle ajusté avec Keras Tuner. Le modèle atteint une précision de 0.90 (courbe orange) qui est en légère baisse par rapport à l'entraînement du modèle de base présenté précédemment. Néanmoins, la fonction de perte a une tendance décroissante et atteint une valeur plus basse et se stabilise autour de 0.30 contre 0.40 pour le modèle de base, **ce qui indique qu'il y a moins de sur-apprentissage avec ces hyperparamètres**.

	Precision	Recall	F1 Score	Support
COVID	0.94	0.93	0.94	204
Lung Opacity	0.89	0.83	0.86	181
Normal	0.78	0.88	0.83	165
Viral Pneumonia	0.98	0.95	0.96	170
Precision			0.90	720
Macro avg	0.90	0.90	0.90	720
Weighted avg	0.90	0.90	0.90	720

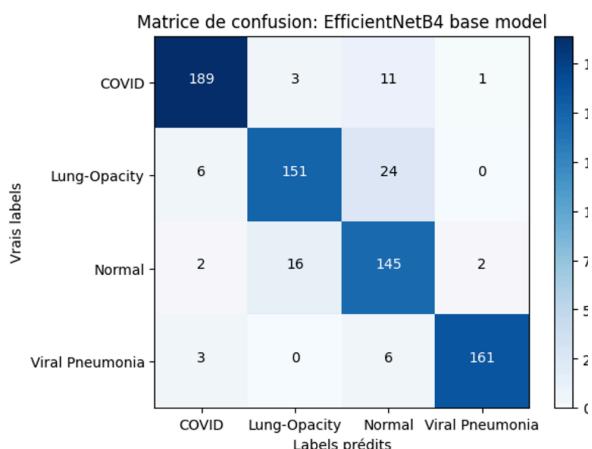


Figure 40. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle EfficientNetB4 ajusté avec keras-tuner

Malgré la légère baisse de précision de l'ensemble validation durant l'entraînement, le modèle atteint une meilleure précision avec les données de l'ensemble test (validation). Le modèle reconnaît mieux la catégorie **COVID** (classe 0) 0.93 contre 0.87 précédemment, et une légère baisse de précision. Finalement le F1-score est légèrement meilleur pour la classe **COVID**, la classe qui nous intéresse dans cette étude.

Les résultats présentés ici sont tous issus du modèle EfficientNetB4 qui a donné les meilleures performances suite à un *benchmark* d'entraînement des modèles EfficientNet. Les poids sont ceux issus de l'entraînement sur la base ImageNet. **Il serait intéressant de réentraîner les couches du modèles sur notre jeu de données pour voir s'il y a une amélioration des performances** sachant que les coûts de calcul seront plus conséquent.

6.1.4 Dégelage des couches entraînables de l'EffcientNet B4

Dans cette partie, nous nous intéresserons aux couches entraînables du modèle pour faire un entraînement complet sur le jeu de données que nous avons. En *transfer learning* chaque couche du réseau de neurones a un argument `trainable` (entraînable), ce paramètre permet de distinguer les poids figés, ne sont pas modifiables, des poids qui peuvent être ajustés sur un nouveau jeu de données, ce qui nous intéresse pour faire cette partie.

En outre, les résultats précédents montrent que le modèle peut faire du sur apprentissage, ce qui entraîne une baisse de la précision et des performances en général. Une des solutions possible est d'ajouter des données d'entraînement, il y a des techniques pour générer des données supplémentaires dans le cas où le jeu de données est très petit, dit de *data augmentation* (ou augmentation de données) en faisant des transformations simples sur le jeu de données initial. L'autre possibilité est d'ajouter un peu plus de données si l'on a à disposition, notre cas ici. Pour ce qui suit, **on prendra 1345 images par classe**.

En ce qui concerne la construction du modèle, le modèle de base est couplé avec une couche `GlobalAveragePooling2D()` et une couche `Dense()` pour faire les prédictions de nos quatre classes. Toutes les couches avec poids ajustables sont entraînées. **La compilation et l'entraînement du modèle se font avec les mêmes paramètres que les parties précédentes.**

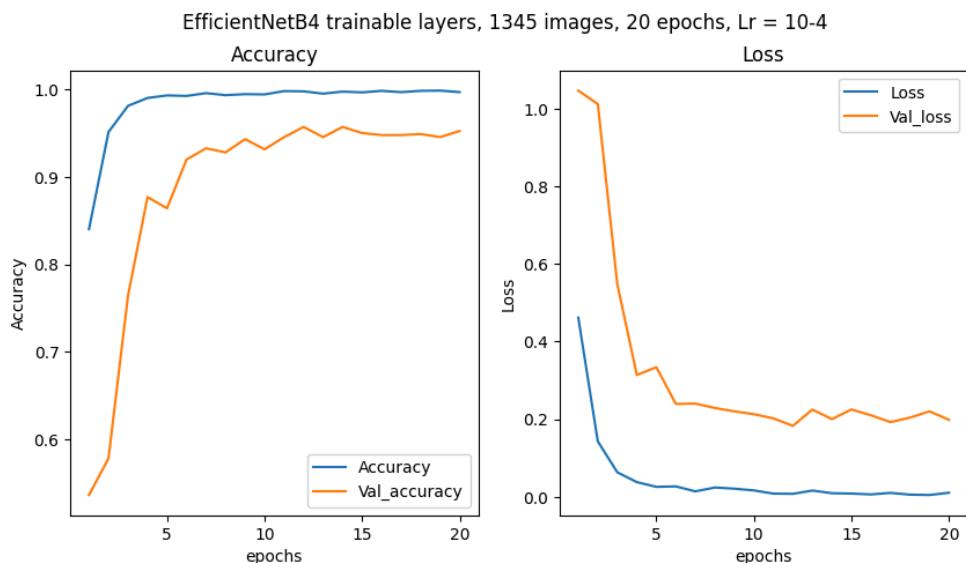


Figure 41. Courbe d'apprentissage du modèle EfficientNetB4 avec les couches entraînables

Comparé aux entraînements précédents, il y a une nette amélioration en entraînant les couches du modèle. La précision de l'entraînement (sur le set de validation de l'ensemble entraînement, courbe orange) atteint 0.95 et la perte se stabilise autour de 0.20 (valeur la plus faible jusqu'ici). Par ailleurs, il y a **nettement moins de surapprentissage, ceci est certainement dû à l'augmentation du jeu de données (900 images/classe précédemment à 1345 images/classe)**.

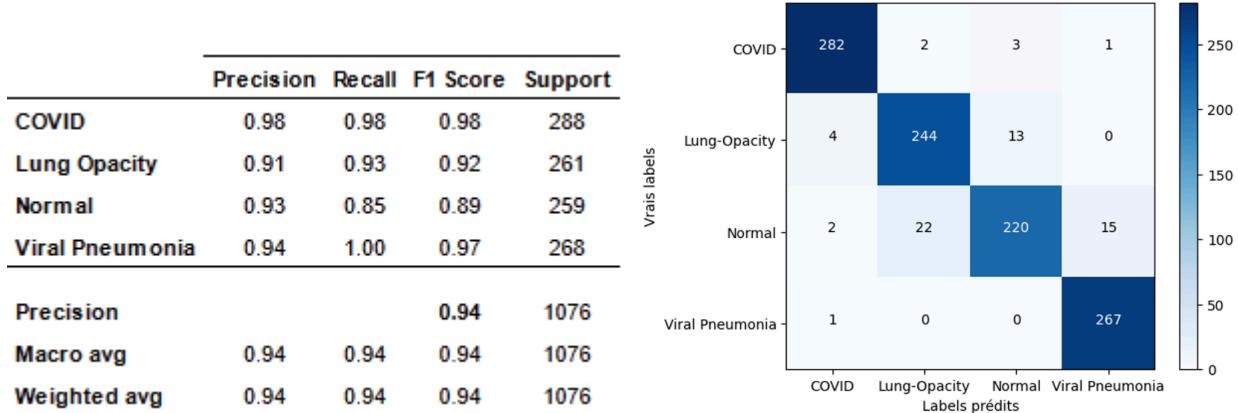


Figure 42. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle EfficientNetB4 ré-entraîné

Avec une précision globale de 0.94, c'est le meilleur modèle que nous avons eu pour cette partie concernant la famille de modèles EfficientNet. De plus, le modèle semble bien plus performant concernant la classe qui nous intéresse ici (classe COVID), avec une reconnaissance des radiographies COVID à 0.98 avec précision. Pour la suite de nos travaux, le meilleur modèle sera adopté et utilisé pour l'interprétabilité et la suite de cette étude.

6.2. VGG16

Au vu des performances solides de VGG16 en matière de *transfer learning* basique, nous avons décidé de poursuivre l'exploration de ses capacités pour notre projet de classification par le biais d'une approche de fine tuning.

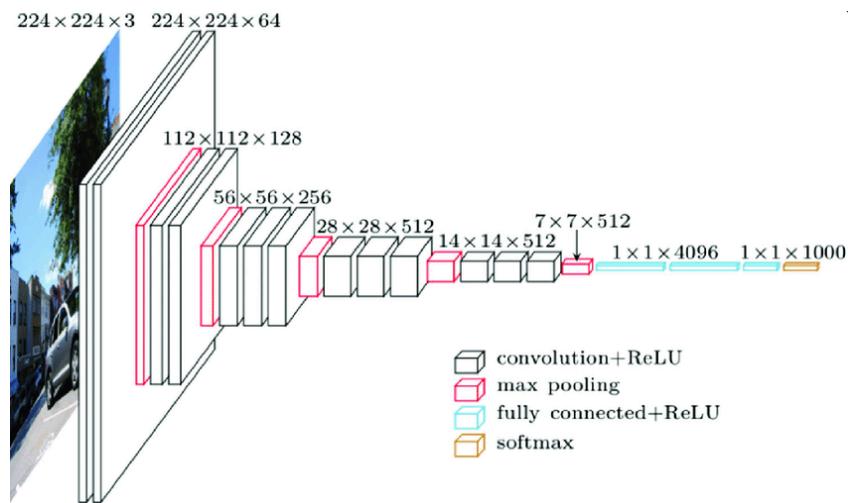


Figure 43. Illustration de l'architecture du modèle VGG16

Ce modèle est l'un des CNN les plus reconnus et efficaces pour la classification d'images. Ses couches convolutives intègrent des filtres de 3x3 avec un pas de 1 et du zero-padding pour maintenir la dimension spatiale intacte. Les couches de max-pooling de 2x2 avec un pas de 2 servent à réduire progressivement la taille des cartes d'activation. Une caractéristique distincte du VGG-16 est l'usage systématique de plusieurs couches convolutives 3x3 en série avant chaque étape de max-pooling, ce qui accroît la profondeur du réseau tout en conservant une complexité raisonnable. Les trois dernières

couches sont entièrement connectées avec respectivement 4096, 4096 et 1000 neurones, cette dernière correspondant aux 1000 classes d'ImageNet, la base sur laquelle le VGG-16 a été formé.

Malgré diverses modifications apportées à l'architecture du modèle (ajout de couches de convolutions et denses avec différents paramètres), nous n'avons pas réussi à améliorer significativement les performances du modèle. Au contraire, une complexification excessive pouvait même dégrader les résultats, tout en augmentant la consommation de ressources. L'intégration d'une couche de dropout juste avant la classification s'est toutefois révélée bénéfique pour limiter le surapprentissage et améliorer les performances globales.

Nous avons également expérimenté le dégel progressif de couches supplémentaires avant leur entraînement pour mieux adapter le modèle à nos données spécifiques. Débloquer les deux derniers blocs de convolutions a offert un bon équilibre entre les résultats et le temps de calcul. Partant de cette base, nous avons utilisé keras-tuner pour rechercher les paramètres d'entraînement optimaux pour la version finale du modèle destinée à notre application. **Les paramètres ajustés lors de ce tuning incluent le nombre de neurones de la dernière couche dense (*unit*), le taux de *dropout* et le taux d'apprentissage (*learning_rate*).** La précision est le critère principal pour évaluer l'efficacité des différentes configurations testées.

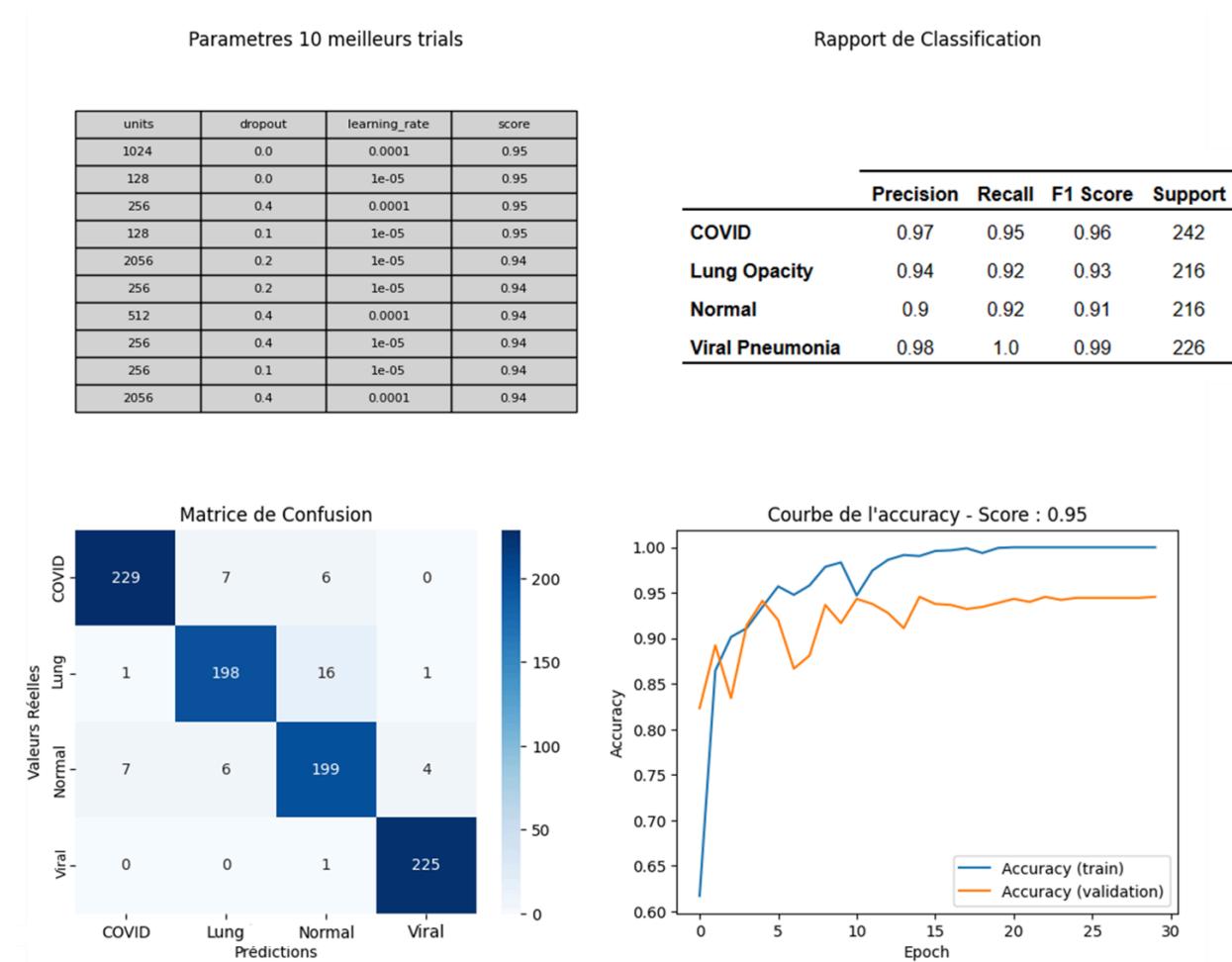


Figure 44. Résultats de classification avec le modèle VGG16 : optimisation des hyperparamètres, évaluation de la précision et analyse des confusions

Après l'utilisation du Kera Tuner il apparaît que nous obtenons les meilleurs scores de précision pour un taux d'apprentissage de 1e-4. Le meilleur modèle semble celui pour lequel la dernière couche dense contient 1024 neurones associée à un **dropout** de 0.0. Il semble donc ici que la présence d'une couche de dropout n'a pas d'impact déterminant sur les performances du modèle.

Le modèle a donc été entraîné avec ces paramètres ce qui nous permet d'améliorer encore l'efficacité du modèle par rapport à ce que nous avions obtenu sans *finetuning*. Les classes les mieux prédites sont **COVID** et **Viral (Viral pneumonia)**, suivies par les classes **Lung Opacity** et **Normal**. De façon intéressante, toutes nos métriques sont au-dessus de 90% et suite à l'entraînement du modèle avec les meilleurs paramètres nous obtenons une accuracy globale de 95%. Ce modèle semble donc capable de fournir des résultats plus qu'acceptables tout en ayant un coût computationnel très contenu.

6.3. ResNet

ResNet, ou *Residual Network*, est un modèle de réseau neuronal profond couramment utilisé dans le domaine de la vision par ordinateur. Il a été introduit par les travaux de Kaiming He et al. (2015).

L'idée principale derrière ResNet est d'introduire des connexions résiduelles, ou *skip connections*, qui permettent au réseau de sauter des couches, facilitant ainsi l'entraînement de réseaux beaucoup plus profonds. Ces connexions résiduelles sont implémentées en ajoutant une sortie de couche à l'entrée de cette même couche. Cela permet de résoudre le problème de la dégradation de la performance du réseau avec l'augmentation de sa profondeur, en permettant à l'optimiseur de se concentrer sur l'apprentissage des résidus (différences entre l'entrée et la sortie de la couche) plutôt que d'apprendre des transformations directes.

La caractéristique principale de ResNet est l'utilisation de blocs résiduels. Ces blocs sont constitués de plusieurs couches convolutionnelles, généralement avec des filtres de petite taille (3x3), suivies d'une fonction d'activation ReLU. La sortie de ces couches est ajoutée à l'entrée du bloc (d'où le nom "résiduel"), et la sortie est passée à travers une autre fonction d'activation ReLU. Ces blocs peuvent être empilés pour former un réseau très profond.

Le premier modèle testé a été le ResNet 50. Montrant un résultat inférieur au LeNet-5, même après transfer learning et dégel des deux dernières couches de neurones, il a été écarté.

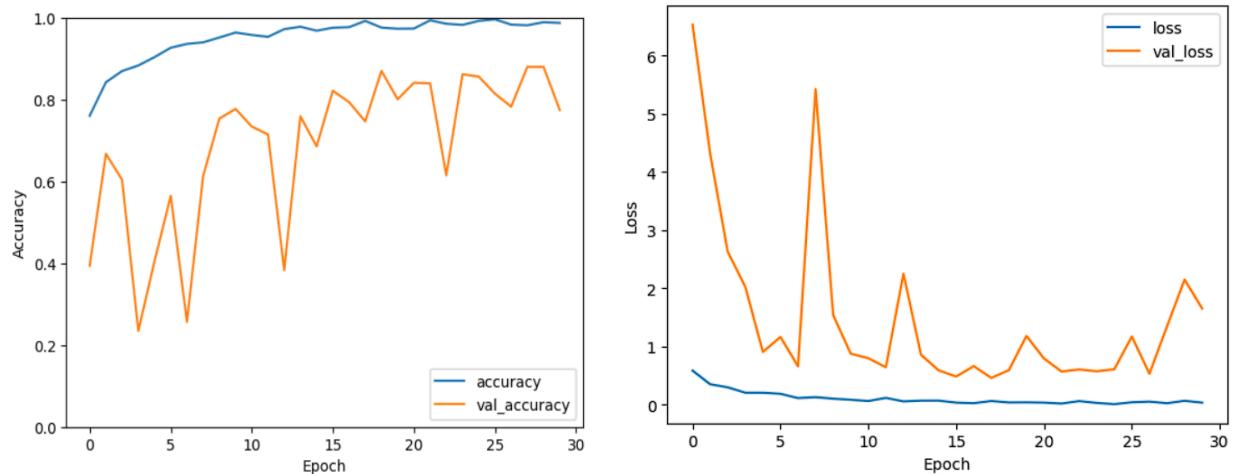


Figure 45. Courbe d'apprentissage du modèle ResNet50 avec un paramétrage fin

	Precision	Recall	F1 Score	Support
COVID	0.88	0.65	0.75	204
Lung Opacity	0.63	0.81	0.71	181
Normal	0.71	0.82	0.76	165
Viral Pneumonia	0.98	0.84	0.91	170
Precision			0.78	720
Macro avg	0.78	0.78	0.78	720
Weighted avg	0.78	0.78	0.78	720

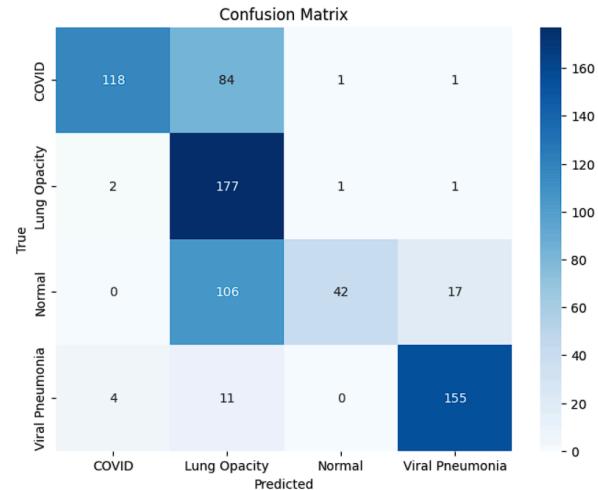


Figure 46. Rapport de classification du modèle ResNet50 avec un paramétrage fin

De la même façon que les modèles ResNet en général, ce premier modèle a cédé au surapprentissage. Les modèles ResNet testés, à contrario d'autres modèles précédemment vus, ne contiennent pas de *dropout*.

Le *dropout* comme nous l'avons vu précédemment, est une technique de régularisation couramment utilisée dans les réseaux neuronaux pour prévenir le surapprentissage en désactivant aléatoirement un certain pourcentage des neurones pendant l'entraînement. Néanmoins, le *dropout* n'est pas une caractéristique standard de l'architecture ResNet telle que définie dans les travaux originaux de Kaiming He et al. (2015). **Le *dropout* n'est généralement pas utilisé dans les architectures ResNet de base.**

Le modèle ResNet le plus performant a été le InceptionResNetV2. Afin de faire fonctionner ce modèle, un redimensionnement des images (299x299x3) a été nécessaire et est introduit dans la phase de preprocessing.

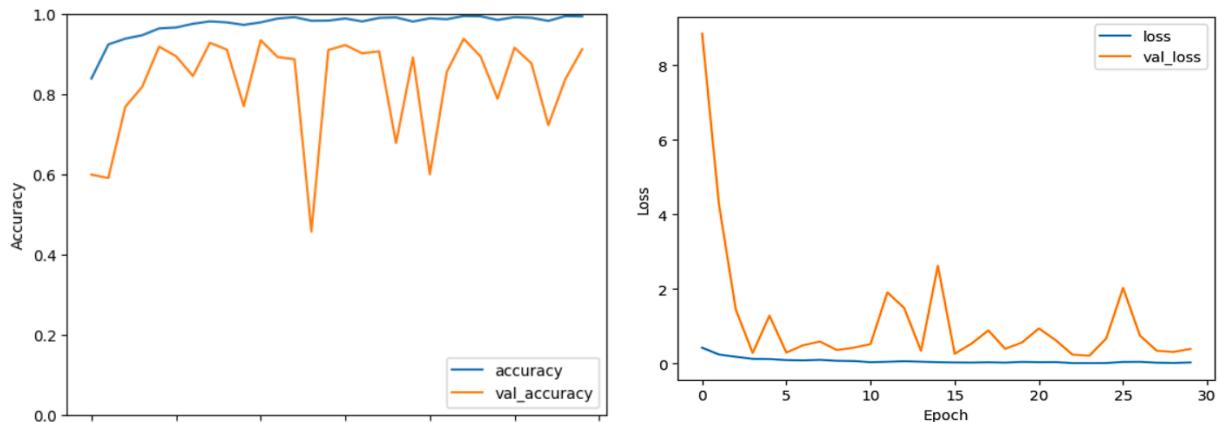


Figure 47. Courbe d'apprentissage du modèle InceptionResNetV2 avec un paramétrage fin

Avec une accuracy atteignant 0.91 pour notre modèle de classification, **le modèle est prometteur mais reste moins performant et plus instable que VGG16 et EfficientNet.**

	Precision	Recall	F1 Score	Support
COVID	0.96	0.97	0.96	288
Lung Opacity	0.84	0.93	0.88	261
Normal	0.91	0.76	0.83	259
Viral Pneumonia	0.94	0.99	0.96	268
Precision			0.91	1076
Macro avg	0.91	0.91	0.91	1076
Weighted avg	0.91	0.91	0.91	1076

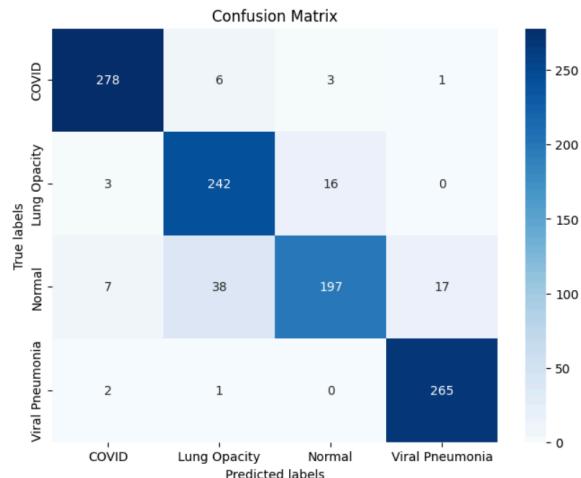


Figure 48. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle InceptionResNetV2 avec un paramétrage fin

Bien que performant, le modèle tend à être freiné dans ses performances par la classe **Lung_Opacity**, dans laquelle il classe des poumons sains et vice-versa.

Quelques poumons sains sont aussi incorrectement classés en **Viral Pneumonia**.

Ce modèle est donc performant, et supprimer la classe Lung_Opacity bénéficierait certainement beaucoup au InceptionResNetV2.

6.4. DenseNet

L'architecture du modèle DenseNet201 en fait un modèle particulièrement efficace pour la classification d'images. En effet, il présente une connectivité dense entre les couches, chaque couche qui est connectée à toutes les couches situées en aval dans un bloc. Un bloc Dense est composé de plusieurs couches de convolution suivies d'une couche de normalisation et d'activation. Les sorties de toutes les couches précédentes sont concaténées et servent d'entrée pour les couches suivantes.

Pour réduire le coût en ressources computationnelles, le modèle utilise des couches *bottleneck* (de rétrécissement) avant chaque couche de convolution. Ces couches permettent de réduire le nombre de canaux en entrée avant d'y appliquer la convolution.

Entre les blocs Dense, plusieurs couches de transition réduisent la résolution spatiale des activations en utilisant des convolutions et un sous-échantillonnage, ce qui permet de réduire la complexité du modèle. C'est un *MaxPooling* ou un *AveragePooling* qui sont généralement employés pour réduire progressivement la taille des activations et extraire les caractéristiques les plus importantes. A noter que dans la plupart des implémentations, la fonction d'activation ReLU est utilisée après chaque couche de convolution pour introduire une non-linéarité dans le modèle. La sortie du réseau est classiquement une couche de classification softmax qui produit les probabilités pour chaque classe d'objet.

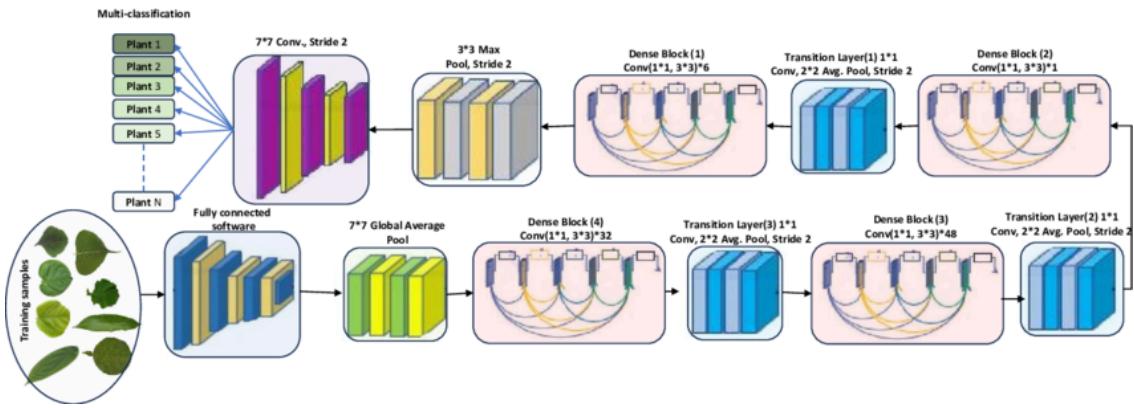


Figure 49. Illustration de l'architecture d'un bloc d'extraction de features du modèle DenseNet201

Les résultats via le *transfer learning* ont montré une efficacité intéressante. Pour cette raison, le modèle a été testé avec plusieurs adaptations pour notre problématique.

Dans un premier temps, le **dégel des dernières couches du modèle permet au réseau de s'adapter plus efficacement aux caractéristiques spécifiques des radiographies**, nécessitant cependant une ressource computationnelle plus importante. Les dernières couches du réseau capturent des informations de plus haut niveau, qui peuvent être cruciales pour la classification précise des différentes conditions pulmonaires associées à la COVID-19. **Nous avons testé jusqu'aux 2 derniers blocs, mais les ressources nécessaires devenaient très importantes pour un gain de performance jugé non-significatif.**

Ainsi, plusieurs adaptations ont été appliquées au modèle DenseNet201 :

- Le **dégel a été conservé pour les 32 dernières couches du modèle**, soit l'équivalent d'un bloc.
- Afin de mapper les caractéristiques extraites par le réseau aux classes de classification spécifiques (dans ce cas, quatre classes de radiographies), une **tête de classification est ajoutée** au-dessus de la sortie du modèle.
- Une couche de **GlobalAveragePooling2D()** est utilisée pour agréger les caractéristiques spatiales en une seule valeur par canal, ce qui réduit la dimensionnalité des données et permet de **limiter le surapprentissage**. De plus, cette couche est préférée dans notre problématique car elle permet de conserver la structure spatiale tout en réduisant la dimensionnalité des données.
- Une couche **Dense()** de 256 neurones avec une fonction d'activation ReLU est ajoutée pour **introduire de la non-linéarité et extraire des caractéristiques discriminantes supplémentaires**. La **régularisation L2 avec un coefficient de 0.01 est appliquée via cette même couche pour réduire le surapprentissage** en ajoutant une pénalité sur les poids du réseau.
- Une couche de **BatchNormalization()** est utilisée pour normaliser les activations de la couche précédente, ce qui **accélère la convergence du modèle et réduit la sensibilité à l'initialisation** des poids de la couche précédente.
- Une couche de **Dropout()** avec un taux de 0.5 est ajoutée pour **régulariser le modèle** en désactivant aléatoirement la moitié des neurones pendant l'entraînement, ce qui permet, là encore, de réduire le risque de surapprentissage.
- Une dernière couche **Dense()** avec une activation **softmax** est utilisée comme couche de sortie pour **produire les probabilités pour chacune des quatre classes de radiographies**.

La combinaison du dégel des dernières couches du modèle avec l'ajout d'une tête de classification adaptée à la problématique que nous avons permis alors **d'optimiser l'architecture afin d'obtenir des performances élevées et une généralisation efficace aux données de validation**.

A noter que l'ajout d'une couche `Flatten()` n'a pas été nécessaire car après application du `GlobalAveragePooling()`, les caractéristiques spatiales sont réduites à une seule valeur par canal plutôt que d'être aplatis en un vecteur unidimensionnel.

Afin de limiter le surapprentissage et par la même occasion réduire le temps d'entraînement, un *callback* `EarlyStopping()` a été utilisé afin de surveiller la perte de validation (`val_loss`) via un seuil de 1e-3 sur cinq époques consécutives. Le maximum d'images possible sans déséquilibrer les classes a été utilisé (1345 images par classe) avec un `batch_size = 32` et le `shuffle = True` afin de mélanger les images à chaque époque.

	Precision	Recall	F1 Score	Support
COVID	0.98	0.98	0.98	288
Lung Opacity	0.95	0.88	0.91	261
Normal	0.87	0.95	0.91	259
Viral Pneumonia	1.00	0.99	0.99	268
Precision			0.95	1076
Macro avg	0.95	0.95	0.95	1076
Weighted avg	0.95	0.95	0.95	1076

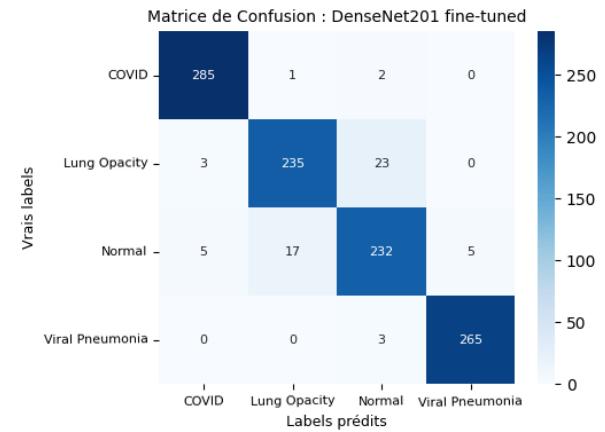


Figure 50. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle DenseNet201 avec un paramétrage fin

Le rapport de classification montre des valeurs élevées pour la précision, le rappel et le F1 Score pour chaque classe ce qui indique que le modèle est particulièrement performant dans la distinction entre les différentes conditions. A noter cependant qu'il performe tout particulièrement dans la distinction de la classe **COVID** et de la classe **Viral Pneumonia** mais est un peu moins efficace dans la détection des classes **Normal** et **Lung_Opacity**. Pour le **COVID**, le modèle a très bien performé, avec seulement 3 faux positifs et faux négatifs. Les résultats pour les autres conditions sont également bons, mais on note quelques erreurs, par exemple, 23 cas de **Lung_Opacity** ont été confondus avec la classe **Normal**. Néanmoins, ces erreurs semblent être faibles en comparaison avec le nombre total de prédictions correctes.

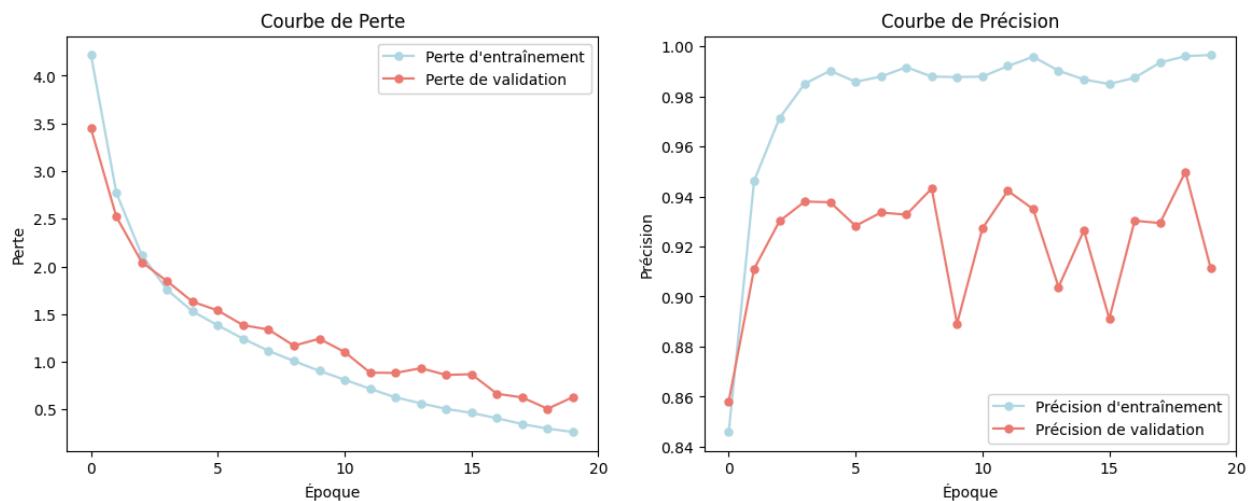


Figure 51. Courbes d'apprentissage et de perte du modèle ResNet152V2 avec un paramétrage fin

Les courbes de perte indiquent que la perte de validation diminue et semble converger avec la perte d'entraînement, **ce qui suggère que le modèle ne surapprend pas (overfitting) de manière significative**. C'est un bon signe que le modèle généralise bien sur des données non vues.

Les courbes de précision montrent une haute précision pour l'entraînement qui reste stable après quelques oscillations initiales, tandis que la précision de validation montre plus de variabilité. **Cela pourrait indiquer que le modèle pourrait encore être amélioré si plus d'époques seraient sélectionnées par exemple.**

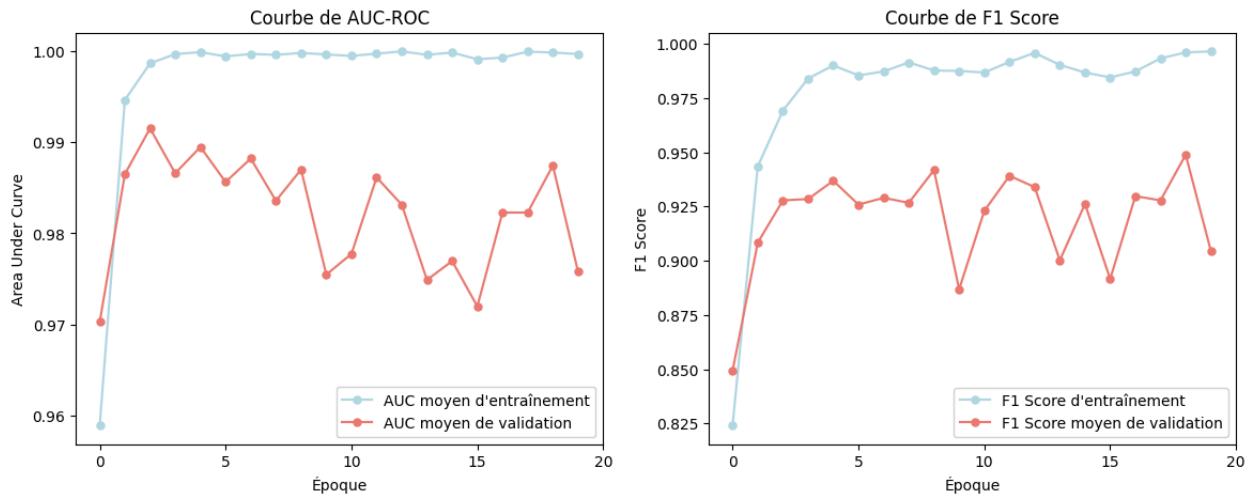


Figure 52. Courbes AUC-ROC et de F1 Score du modèle ResNet152V2 avec un paramétrage fin

Les valeurs AUC élevées indiquent que le modèle possède une bonne capacité à distinguer entre les différentes classes. Cela confirme la performance du modèle suggérée par les précédentes métriques de précision, rappel, et score F1. Plus précisément, une courbe AUC-ROC élevée suggère que le modèle a un bon taux de vrais positifs tout en maintenant un faible taux de faux positifs. L'AUC moyen sur les données d'entraînement reste relativement stable et élevé, ce qui indique que le modèle a appris à bien classer les échantillons d'entraînement au fil des époques. **Cependant, la ligne représentant l'AUC moyen sur les données de validation montre plus de variation, indiquant que la performance du modèle sur les données de validation est moins stable.** Bien que les valeurs restent élevées, cette variation pourrait signifier que le modèle aurait encore du potentiel pour être amélioré.

L'écart entre les F1 Scores d'entraînement et de validation pourrait indiquer un léger surapprentissage. Bien que le modèle soit efficace sur les données d'entraînement, il semble avoir un peu plus de mal à généraliser ses prédictions avec la même précision sur les données de validation. De plus, les variations des F1 Scores de validation suggèrent que le modèle pourrait être sensible à certaines spécificités des données de validation et que cela peut être le résultat d'une variété de facteurs, comme une différence notable dans la qualité de certaines données de validation ou la présence de caractéristiques difficiles à apprendre. **Cependant, malgré cela, il ne semble pas y avoir de tendance à la baisse ou à l'augmentation du F1 Score de validation au fil du temps, cela pourrait suggérer que le modèle a atteint ses capacités d'apprentissage avec cette configuration et les données fournies.**

7. Interprétabilité du modèle final

7.1. Présentation du modèle final sélectionné

Le modèle VGG16 semble être le meilleur modèle que nous ayons pu produire après grâce au *finetuning*. En reprenant le rapport de classification et la matrice de confusion nous pouvons observer que :

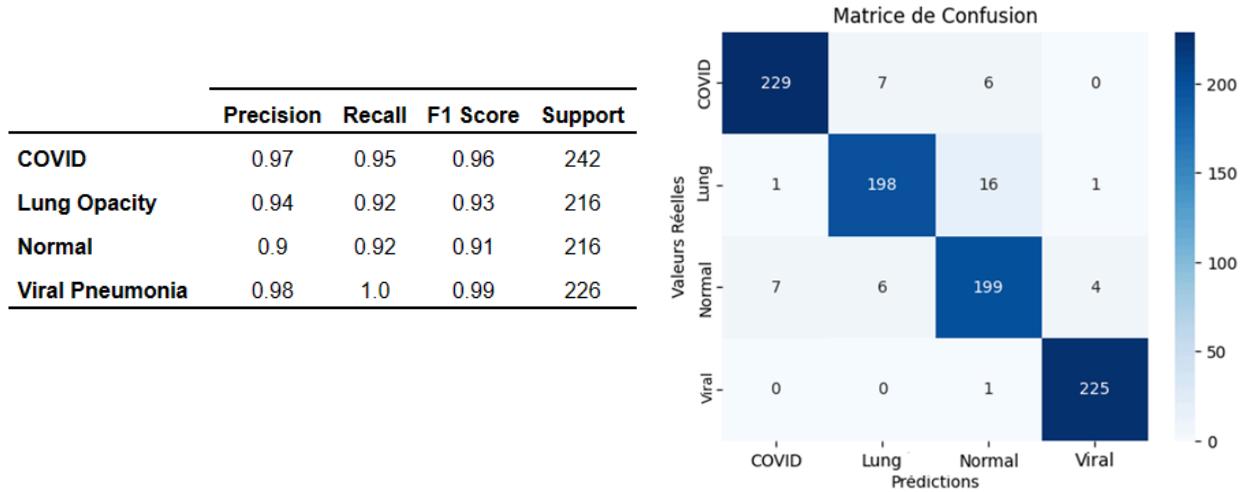


Figure 53. Rapport de classification (à gauche) et matrice de confusion (à droite) du modèle VGG16 optimisé avec un paramétrage fin

Le modèle a une précision générale élevée (score de 0.95 sur les données de validation), ce qui est corroboré par les scores F1 élevés pour chaque classe.

La classe **COVID** est prédictive avec une précision et un rappel très satisfaisants, ce qui est crucial dans un contexte médical où manquer un cas positif pourrait être très risqué.

Il y a cependant quelques erreurs de classification entre les classes **Normal**, **COVID** et **Lung_Opacity**, comme le montre la matrice de confusion. Par exemple, 16 classes **Lung_Opacity** ont été classées à tort comme **Normal** et 7 classes **Normal** ont été classées à tort comme **COVID**.

Sur la base de ces métriques, le modèle semble très performant, surtout pour les classes **Virale Pneumonia** et **COVID**, avec un rappel performant et une précision très élevée. Cependant, pour les autres classes (**Normal**, **Lung_Opacity**), bien que les performances soient élevées, il y a eu quelques confusions, comme indiqué par les scores de rappel et de précision légèrement inférieurs.

7.2. Interprétabilité

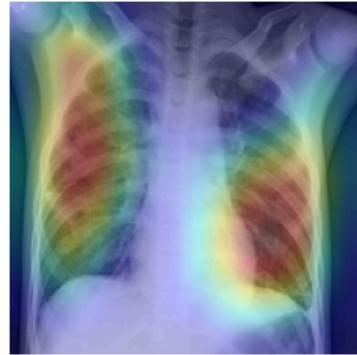
Afin d'étudier la façon dont notre modèle analyse les images que nous lui fournissons, nous procédons à une visualisation par **Grad-CAM** (*Gradient-weighted Class Activation Mapping*) afin d'observer les zones de l'image qui ont été les plus importantes pour la prédiction effectuée par le modèle.

Dans un premier temps nous affichons un jeu d'images choisies aléatoirement. **Nous pouvons observer que le modèle choisit de se baser sur la zone des poumons et qu'il se focalise parfois sur les annotations présentes sur les radiographies.** Malgré cela, il semble parvenir à classer correctement les images.

Classe : Normal, Prédiction : Normal



Classe : Viral, Prédiction : Viral



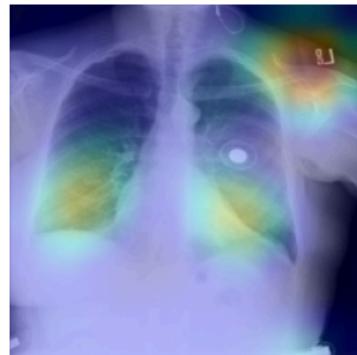
Classe : COVID, Prédiction : COVID



Classe : Viral, Prédiction : Viral



Classe : Normal, Prédiction : Normal



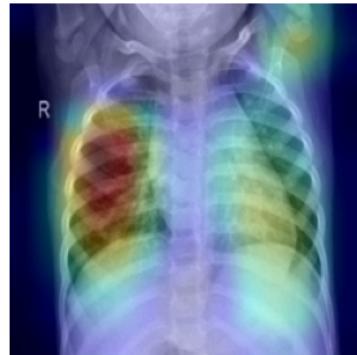
Classe : Viral, Prédiction : Viral



Classe : Viral, Prédiction : Viral



Classe : Normal, Prédiction : Viral



Classe : Lung, Prédiction : Lung

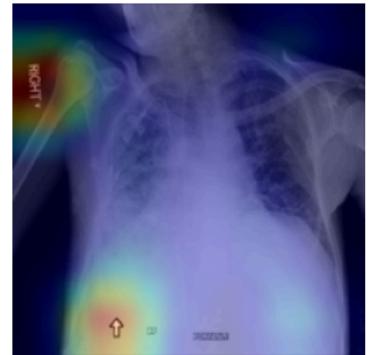


Figure 54. Sorties aléatoires de GRAD-CAM pour 9 prédictions correctes

Dans un second temps nous affichons de façon aléatoire 10 images parmi les mauvaises prédictions effectuées par le modèle. Dans ce cas de figure, **il semble que nous voyons apparaître une surreprésentation des images pour lesquelles le modèle s'est concentré sur les annotations des radiographies.**

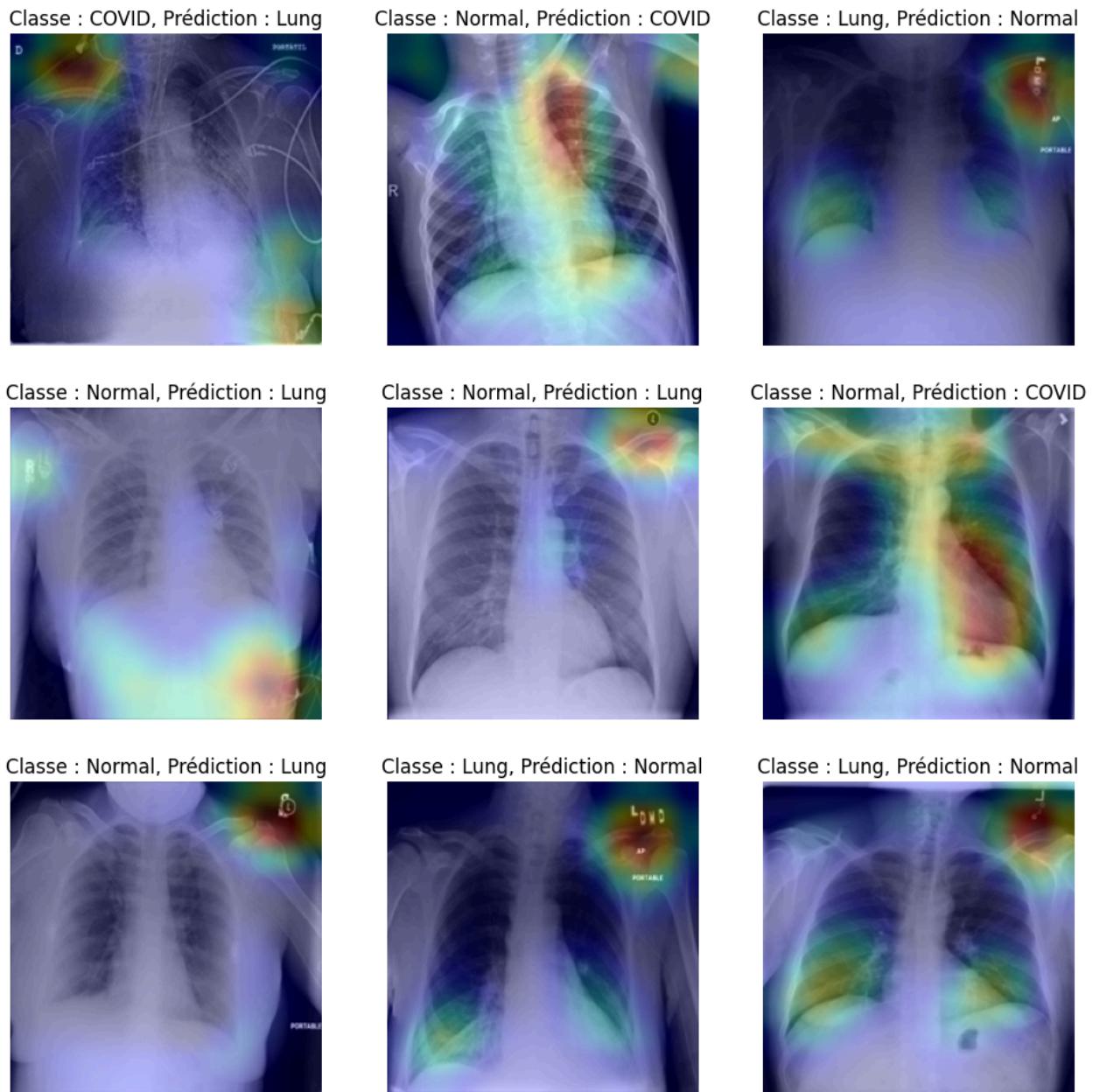


Figure 55. Sorties aléatoires de GRAD-CAM pour 9 prédictions incorrectes

7.3. Qualités et limites

7.3.1. Qualités du modèle et capacité de prédiction

Le travail réalisé pour le paramétrage fin du modèle a permis d'en faire **ressortir des performances plus que satisfaisantes pour notre problématique**. Les résultats sont par ailleurs les meilleurs que nous avons pu avoir, en prenant en compte l'ensemble des modèles que nous avons tenté.

En effet, la particularité de notre modèle réside dans des **performances de prédictions équilibrées et satisfaisantes entre les différentes classes** tout en étant relativement **léger et réactif lorsqu'une prédiction lui est demandée**. Ces caractéristiques rendent ce modèle déployable sur des plateformes avec des ressources computationnelles limitées.

La particularité de notre modèle réside dans une **sensibilité (aucun faux négatif pour la même classe)** et une **spécificité (pas de faux positifs pour la classe COVID)** élevées, en particulier pour les deux classes pour lesquelles nous avons le plus d'intérêt : **COVID et Normal** avec respectivement 0.96 et 0.91 de F1 Score, démontrant que la modèle est capable de différencier efficacement entre les catégories.

Afin de valider notre stratégie de recherche d'hyper paramètres et d'entraînement de modèle nous avons également procédé à un entraînement en utilisant trois jeux de données distincts (Entrainement, Test, Validation) afin de s'assurer d'avoir un jeu d'images jamais vu pour notre modèle.

Nous avons obtenus sensiblement les mêmes résultats que précédemment en procédant ainsi, aussi bien au niveau des métriques que le comportement du modèle au vis à vis des Grad-CAM :

VGG16	Precision	Recall	F1 Score	Support
COVID	0.96	0.96	0.96	206
Lung Opacity	0.91	0.89	0.9	182
Normal	0.9	0.91	0.91	199
Viral Pneumonia	0.98	0.98	0.98	213
Precision			0.94	800
Macro avg			0.93	800
Weighted avg			0.94	800

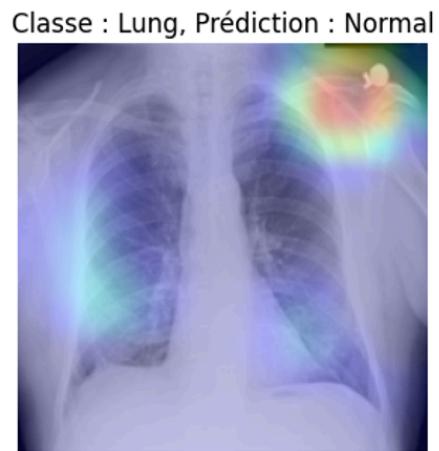
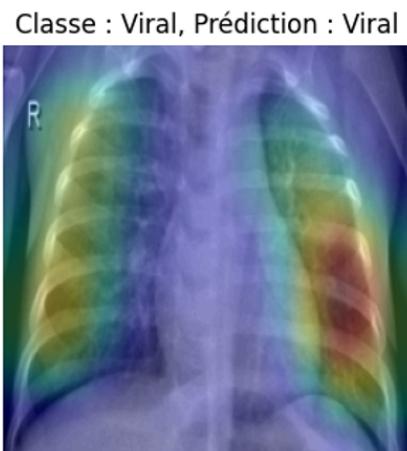
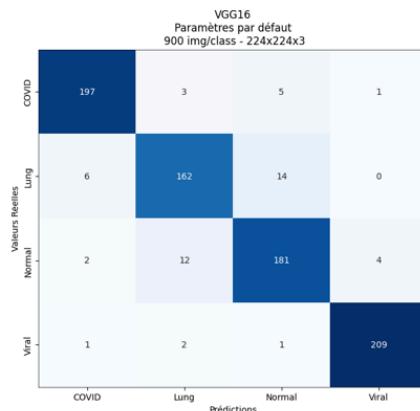


Figure 56. Evaluation des performances du modèle sur un ensemble de validation après entraînement

En détail, le modèle VGG16 possède une architecture profonde et des couches convolutives de petite taille (3x3), ce qui lui permet de récupérer des features précises tout en maintenant le contrôle sur le nombre de paramètres. De multiples couches dont notamment le *MaxPooling* permettent de concentration l'information et ainsi déjà participer à la réduction du risque de surapprentissage.

Malgré notre tentative pour prévenir le surapprentissage, le modèle sélectionné s'avère ne pas avoir besoin d'une couche de `Dropout()`. Ceci nous indique donc que la quantité de données utilisée et l'architecture profonde de ce modèle finement ajusté permettent déjà de contrôler suffisamment le surapprentissage.

L'usage du Keras Tuner pour optimiser les hyperparamètres nous a permis de trouver la meilleure configuration avec un choix de 1024 neurones dans la dernière couche `Dense()`, un taux d'apprentissage de 1e-4 et le retrait de la couche de `Dropout()` initialement ajoutée. **Ces paramètres optimaux nous permettent d'avoir un modèle apprenant des features discriminantes tout en évitant le surajustement à notre ensemble d'entraînement.**

Un autre avantage indéniable de ce modèle est son **équilibre entre ses performances et son coût en ressources computationnelles**. Débloquer progressivement les deux derniers blocs de convolutions nous a permis de porter une attention particulière à l'équilibre entre les besoins en ressources et les performances du modèle, très important dans un cadre opérationnel.

7.3.2. Tests complémentaires et pistes d'améliorations

Comme nous l'avons vu précédemment, notre modèle prédictif est très efficace sur les classes **COVID** et **Viral Pneumonia**. **Cependant, à la vue des métriques associées, un potentiel d'amélioration est toujours présent** sur ces classes, mais aussi pour les deux autres classes présentes dans notre dataset.

Pour cela, nous avons creusé quelques pistes d'amélioration qui semblent intéressantes.

- **Qualité des images**

Nous nous sommes questionnés sur la qualité du dataset. Dans un premier temps, avoir des images différentes avec des variations plus-ou-moins importantes pouvait forcer nos modèles à extraire des features particulières sur des éléments non pertinents des radiographies (comme du texte, des dates, des logos, etc.). Ainsi, nous avons testé en créant un dataset propre comprenant 4800 images (1200 images par classe). Pour cela, nous avons réalisé une extraction aléatoire de 1345 images par classe, puis nous avons nettoyé à la main la sélection en supprimant les images jugées non-pertinentes (radiographies incomplètes, grande quantité de textes sur l'image, annotations et trop forte/faible luminosité) en veillant à bien conserver l'équilibre des images dans chaque classe.

Radiographies “propres”



Radiographies “non pertinentes”

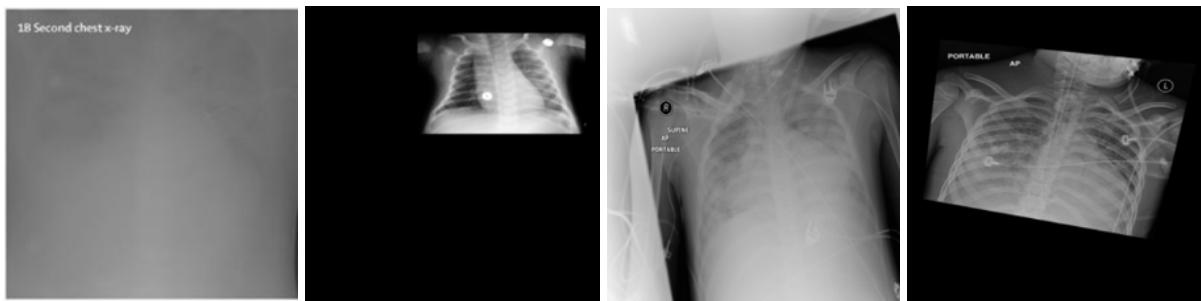


Figure 57. Exemples de radiographies jugées ‘propres’ et ‘non pertinentes’

Après plusieurs tests sur l'EfficientNet et le VGG16, les résultats ne semblent pas avoir variés de manière importante, pire, les résultats semblent un peu moins bons avec une précision de l'ensemble de validation moyenne descendue à 0.89.

Il est possible qu'avoir retiré de la variabilité et standardiser les images aient pu entraîner de la difficulté au modèle à généraliser en retirant des features importantes.

Pour tester cette hypothèse, nous avons gardé notre jeu de données propre, mais nous avons créé des transformations uniquement sur l'ensemble d'entraînement à l'aide de `ImageDataGenerator()`, notamment :

- une rotation aléatoire dans la plage de [-40 ; 40] degrés,
- un décalage horizontal et vertical aléatoire jusqu'à 20%,
- un cisaillement aléatoire dans le sens antihoraire dans la plage [0 ; 0.2] radians,
- un zoom aléatoire jusqu'à 20%,
- pas de retournements horizontaux,
- remplissage des nouveaux pixels par la valeur du pixel le plus proche de la position d'origine.

Malgré ces transformations, après plusieurs tests sur l'EfficientNet et le VGG16, les résultats n'ont pas montré une progression significative des résultats et restent inférieurs aux résultats sur le jeu de données original. Il est cependant probable que continuer d'améliorer les générateurs d'images avec d'autres transformations, plus-ou-moins importantes pourraient à terme optimiser la généralisation des modèles (Ebenezer, 2022), et les pousser à extraire des features pertinentes sur la base des poumons sur les radiographies.

- Amélioration des masques

Lors de l'interprétation de notre modèle, nous avons pu constater que certaines images étaient bien prédites, mais malheureusement, notre réseau de neurones observait des zones non-pertinentes (comme le contour de la radiographie plutôt que les poumons). Ainsi, nous avons cherché à appliquer les masques pour forcer le modèle à se concentrer sur les poumons. **Cependant, nos tests précédents ont montré des résultats plus faibles après application des masques.**

Hors, les masques de bases sont très restreints et se concentrent sur la partie interne des poumons, ce qui est intéressant, **mais fait probablement perdre une importante quantité d'information qui peut être perçue sur les bords et contours des poumons**. Ainsi, nous avons appliqué une dilatation de 20% des masques basiques à notre disposition grâce à la fonction `dilate()` d'OpenCV.

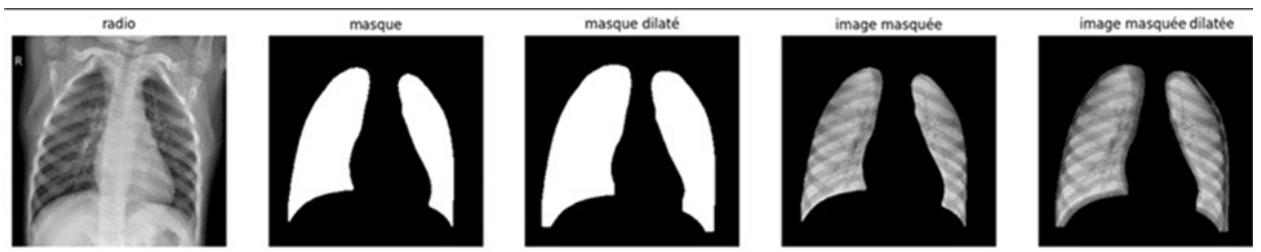


Figure 58. Exemple d'une radiographie avec un masque basique et dilaté à 20%

Après application de ces masques dilatés, les résultats semblent un peu meilleurs qu'avec application des masques basiques, mais restent inférieures à l'utilisation des images sans masque.

VGG16 fine tuned

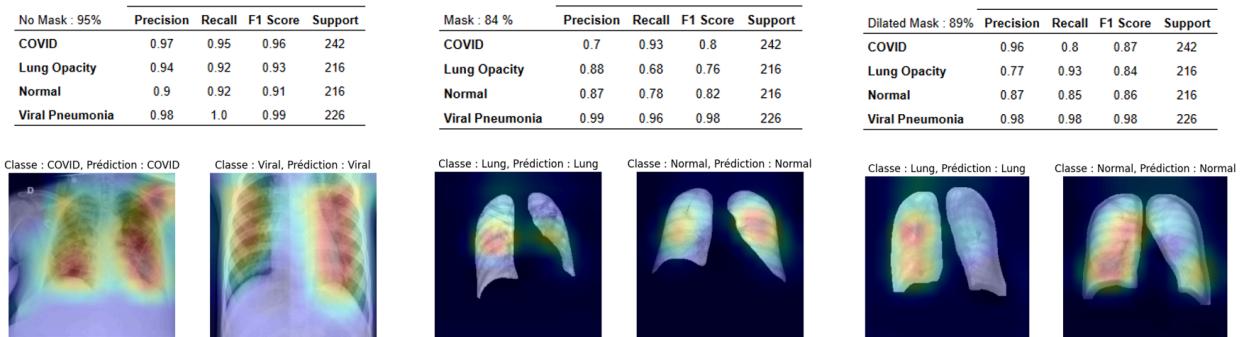


Figure 59. Comparaison des résultats d'apprentissage du modèle VGG16 après finetuning pour des sans masque, avec masque et avec masques dilatés

Il s'agit d'une piste intéressante, il est possible qu'augmenter encore le taux de dilatation des masques jusqu'à qu'il puisse atteindre un seuil à partir duquel plus aucune feature pertinente ne soit détectée.

- **Une architecture plus optimisée**

Nous avons essayé un grand nombre de modèles spécialisés dans la classification d'images qui ont quasiment tous montré des performances très satisfaisantes sur notre jeu de données. Cependant, nous avons pu constater un seuil à partir duquel nous n'avons pas réussi, malgré nos nombreux tests d'amélioration, à augmenter les performances de généralisation de notre modèle.

Ceci peut simplement s'expliquer par une architecture complexe qui n'est peut-être pas complètement adaptée à notre problématique. Il est toujours possible que architecture légèrement différente d'un autre modèle non testé ici puisse s'avérer plus efficace. Ou bien, qu'un modèle spécifiquement entraîné sur des radiographies, à contrario de nos modèles entraînés sur ImageNet, puisse être plus pertinents, mais nous n'avons pas connaissance de tels modèles.

Une autre piste serait **d'utiliser une combinaison de modèles afin d'associer les particularités des différents modèles et augmenter les performances** (avec le VGG, ResNet ou InceptionResNet par exemple) comme cela a été réalisé par Campers (2022) pour classifier des images d'animaux sur pièges photographiques.

- **Des métriques de perte personnalisées**

Nous aurions pu utiliser l'ensemble des données à notre disposition pour augmenter la variabilité et la diversité en radiographies. Cependant, ceci aurait généré un déséquilibre très important entre nos différentes classes. **Instaurer des métriques de perte personnalisées pourraient aider à tenir compte de ce déséquilibre et donner plus de poids aux classes sous-représentées.** Mais nous aurions pu utiliser ces métriques pour accorder plus de poids aux classes **COVID** et **Normal** qui sont principalement les classes que nous cherchions à optimiser dans le cadre de ce projet.

De plus, avec plus de temps, il aurait pu être très pertinent de **questionner des spécialistes du métier pour connaître les types d'erreur les plus critiques dans le diagnostic du COVID-19 à partir de radiographies** et utiliser ces informations dans la conception de métriques de perte personnalisées selon des éléments cliniques significatifs.

La COVID-19 semble présenter des caractéristiques uniques, **telles que des opacités pulmonaires diffuses ou un effet « verre dépoli » dans les radiographies pulmonaires** (Landete et al., 2020 ; Kanne et al., 2021). Des métriques de perte personnalisées auraient pu être conçues pour prendre spécifiquement en compte ces caractéristiques dans le processus de classification.

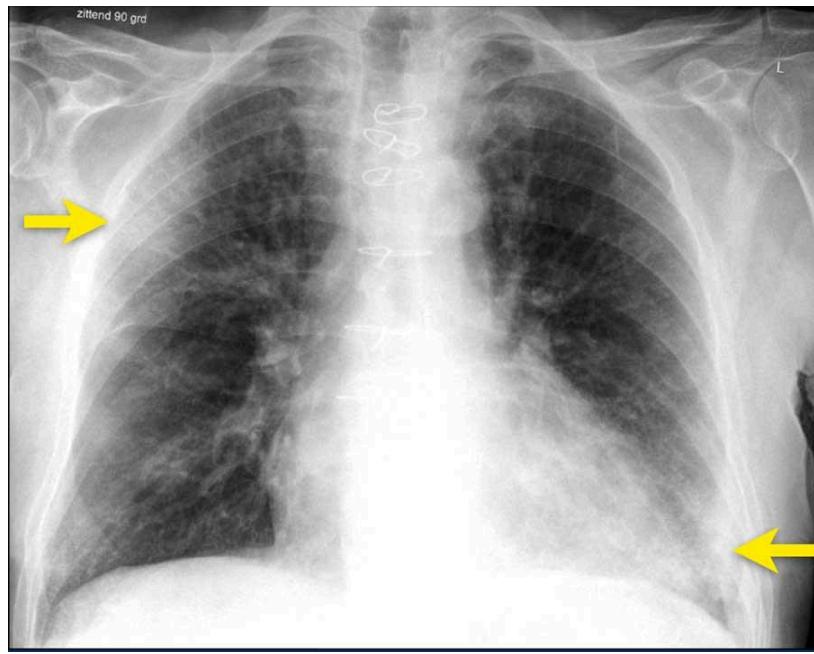


Figure 60. Radiographie du thorax d'un homme de 83 ans souffrant d'insuffisance mitrale, d'hypertension pulmonaire et de fibrillation auriculaire et infecté par le virus COVID-19. Opacification en verre dépoli et consolidation dans le lobe supérieur droit et le lobe inférieur gauche (flèches)⁸

Compte tenu que les métriques personnalisées peuvent être ajustées et adaptées au fur et à mesure que de nouvelles informations sont disponibles ou que les besoins changent, cela aurait probablement permis une meilleure adaptation du modèle et une amélioration continue de ses performances.

⁸ <https://radiologyassistant.nl/chest/covid-19/covid19-imaging-findings>

Bibliographie

- Abdar, A. K., Sadjadi, S. M., Soltanian-Zadeh, H., Bashirgonbadi, A., & Naghibi, M. (2020, November). Automatic detection of coronavirus (COVID-19) from chest CT images using VGG16-based deep-learning. In *2020 27th National and 5th International Iranian Conference on Biomedical Engineering (ICBME)* (pp. 212-216). IEEE.
- Abdullah, S. M. S., & Abdulazeez, A. M. (2021). Facial expression recognition based on deep learning convolution neural network: A review. *Journal of Soft Computing and Data Mining*, 2(1), 53-65.
- An, Y., Hu, T., Wang, J., Lyu, J., Banerjee, S., & Ling, S. H. (2019, July). Lung Nodule Classification using A Novel Two-stage Convolutional Neural Networks Structure'. In *2019 41st annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 6259-6262). IEEE.
- Aziz, A., Attique, M., Tariq, U., Nam, Y., Nazir, M., Jeong, C. W., ... & Sakr, R. H. (2021). An Ensemble of Optimal Deep Learning Features for Brain Tumor Classification. *Computers, Materials & Continua*, 69(2).
- Campers, H. (2022). Automatisation de la reconnaissance d'espèces animales dans des vidéos de pièges photographiques installés dans les forêts tropicales en Afrique centrale, grâce à l'apprentissage profond.
- Darapaneni, N., Subramaniyan, M., Mariam, A., Venkateshwaran, S., Ravi, N., Paduri, A. R., & Gunasekaran, S. (2020, November). Handwritten form recognition using artificial neural network. In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)* (pp. 420-424). IEEE.
- Ebenezer, A. S., Kanmani, S. D., Sivakumar, M., & Priya, S. J. (2022). Effect of image transformation on EfficientNet model for COVID-19 CT image classification. *Materials Today: Proceedings*, 51, 2512-2519.
- Géron, A. (2020). *Deep Learning avec Keras et TensorFlow: Mise en œuvre et cas concrets*. Dunod.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- Islam, M. R., & Matin, A. (2020, December). Detection of COVID 19 from CT image by the novel LeNet-5 CNN architecture. In *2020 23rd International Conference on Computer and Information Technology (ICCIT)* (pp. 1-5). IEEE.
- Jaiswal, A., Gianchandani, N., Singh, D., Kumar, V., & Kaur, M. (2021). Classification of the COVID-19 infected patients using DenseNet201 based deep transfer learning. *Journal of Biomolecular Structure and Dynamics*, 39(15), 5682-5689.
- Kanne, J. P., Bai, H., Bernheim, A., Chung, M., Haramati, L. B., Kallmes, D. F., ... & Sverzellati, N. (2021). COVID-19 imaging: what we know now and what remains unknown. *Radiology*, 299(3), E262-E279.
- Kavi B. Obaid, Subhi R. M. Zeebaree & Omar M. Ahmed (2020). Deep Learning Models Based on Image Classification: A Review. *International Journal of Science and Business*, 4(11), 75-81. DOI: 10.5281/zenodo.4108433
- Kaiming He et al. (2016). Deep Residual Learning for Image Recognition, DOI:10.1109/CVPR.2016.90
- Lagunas, M., & Garces, E. (2018). Transfer learning for illustration classification. arXiv preprint arXiv:1806.02682.

Landete, P., Loaiza, C. A. Q., Aldave-Orzaiz, B., Muñiz, S. H., Maldonado, A., Zamora, E., ... & Couñago, F. (2020). Clinical features and radiological manifestations of COVID-19 disease. *World Journal of Radiology*, 12(11), 247.

LeCun Y., Bengio Y., and Hinton G. Deep learning, *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, DOI: 10.1038/nature14539.

Liew, S. S., Hani, M. K., Radzi, S. A., & Bakhteri, R. (2016). Gender classification: a convolutional neural network approach. *Turkish Journal of Electrical Engineering and Computer Sciences*, 24(3), 1248-1264.

Marques, G., Agarwal, D., & De la Torre Díez, I. (2020). Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. *Applied soft computing*, 96, 106691.

O'Malley, Tom and Bursztein, Elie and Long, James and Chollet, François and Jin, Haifeng and Invenrnizzi, Luca and others (2019). <https://github.com/keras-team/keras-tuner>

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

Villain E. *Utilisation de l'intelligence artificielle pour l'aide au diagnostic des patients atteints de pathologies neuro dégénératives*. (2021). Imagerie. Université Paul Sabatier - Toulouse III. Français.

Wang, Q., Shen, F., Shen, L., Huang, J., & Sheng, W. (2019). Lung nodule detection in CT images using a raw patch-based convolutional neural network. *Journal of digital imaging*, 32, 971-979.

Wenchao Cui , Qiong Lu , Asif Moin Qureshi , Wei Li & Kehe Wu (2020): An adaptive LeNet-5 model for anomaly detection, Information Security Journal: A Global Perspective, DOI: 10.1080/19393555.2020.1797248