

**Rapport de Stage 2A - Utilisation du deep learning pour apprendre à diagnostiquer automatiquement la COVID-19 à partir de CT scans du poumon.**



ENSTA Bretagne  
2 rue F. Verny  
29806 Brest Cedex 9, France

Tony CORREIA DOS REIS (FISE\_2021) <tony.correia@ensta-bretagne.org>

## Résumé

La pandémie de COVID-19 a touché le monde entier de manière significative, que ce soit avec des conséquences directes comme les décès dus à la maladie ou avec des conséquences indirectes comme le chômage dû au confinement obligatoire. Le diagnostic de la maladie est d'une importance cruciale car il permet d'isoler le patient infecté et de ralentir la propagation du virus. Cependant, le processus de test peut prendre beaucoup de temps pour produire des résultats, les tests les plus utilisés sont susceptibles de présenter certaines erreurs et ces tests peuvent être coûteux. Des recherches antérieures ont montré que les caractéristiques visuelles des CT scans du poumon peuvent être associées à la présence de la COVID-19 [4,5]. Par conséquent, ce travail a proposé l'utilisation de réseaux neuronaux convolutifs pour apprendre automatiquement ces caractéristiques visuelles et diagnostiquer si un patient est atteint de la COVID-19. L'un des obstacles à cette approche est la disponibilité publique d'un ensemble de données suffisamment important pour permettre de tirer des enseignements. Ce travail utilise un ensemble de données qui a été produit à partir d'images tirées d'articles scientifiques. Tout d'abord, l'ensemble de données est soigneusement analysé pour identifier les différents types de distorsions et certaines techniques de prétraitement des images sont utilisées pour les traiter. Trois réseaux convolutifs populaires sont testés : VGG16, ResNet-50 et DenseNet-169. Ensuite, des algorithmes de recherche d'hyperparamètres sont utilisés pour obtenir des hyperparamètres qui provoquent une bonne performance. Les résultats sont prometteurs, atteignant des précisions d'environ 80%, mais qui sont encore trop faibles pour l'application. Si davantage de techniques de prétraitement des données étaient utilisées pour nettoyer les données ou si un plus grand ensemble de données moins polluées était disponible, de meilleures performances pourraient sûrement être obtenues.

## Abstract

The COVID-19 pandemic has significantly affected the entire world, be it with direct consequences such as deaths from the disease or from indirect consequences such as unemployment due to mandatory confinement. Diagnosis of the disease is of crucial importance since it allows for the isolation of the infected patient and slows down the spreading of the virus. However, the testing process can take a long time to produce results, the mostly used tests are susceptible to certain errors and these tests can be expensive. Previous research has shown that visual characteristics of lung CT scans can be associated with the presence of COVID-19 [4,5]. Therefore, this work proposes the use of convolutional neural networks to automatically learn these visual patterns and diagnose if a patient has COVID-19. One of the obstacles with this approach is the public availability of a sufficiently large set of data to learn from. This work uses a dataset that was produced from images taken from scientific articles. Firstly, the dataset is carefully analysed to identify the different types of distortions and some image preprocessing techniques are used to deal with them. Three popular convolutional networks are tested: VGG16, ResNet-50 and DenseNet-169. Next, Hyperparameter search algorithms are used to obtain hyperparameters that cause a good performance. The results are promising, reaching accuracies of about 80%, but which is still too low for the application. If more data preprocessing techniques were used to clean the data or if a larger less polluted dataset was available, better performances could surely be obtained.

## Table des matières

Résumé.....	2
Abstract .....	2
Contexte .....	6
Introduction .....	6
1) Analyse des données .....	7
1.2) CT scan (tomographie assistée par ordinateur).....	7
1.3) L'ensemble des données .....	9
1.4) Pollution des données .....	15
1.4.1) Bords blancs .....	15
1.4.2) Inclusion incorrecte de l'angiographie par tomodensitométrie (CTA) .....	15
1.4.3) Annotations.....	16
1.4.4) Distorsion des couleurs .....	17
2) Deep learning .....	18
2.1) Ressources .....	18
2.1.1) Matériel .....	18
2.1.2) Logiciels .....	18
2.1.3) Github .....	18
2.2) Définition du problème comme un problème de machine learning .....	19
2.3) Préparation des données.....	20
2.3.1) Exclusion des images CTA .....	21
2.3.2) Conversion en niveaux de gris .....	21
2.3.3) Normalisation .....	21
2.4) Validation croisée à k plis.....	21
2.4.1) Division de l'ensemble des données .....	22
2.5) Algorithmes de recherche d'hyperparamètres .....	23
2.6) Évaluation du modèle .....	25
2.6.1) Évaluation multicrop.....	25
2.6.2) Visualisation des performances .....	25
2.7) Réduction du surapprentissage .....	27
2.7.1) Augmentation des données .....	27

2.7.2) Régularisation .....	28
2.7.3) Couches dropout .....	28
2.7.4) L'utilisation de modèles pré-entraînés .....	28
2.8) Modèles .....	29
2.8.1) VGG16 .....	30
2.8.2) ResNet-50 .....	31
2.8.3) DenseNet-169 .....	33
2.8.4) Comparaison Grad-CAM .....	35
3) Analyse des résultats .....	36
Conclusion .....	37
Annexes .....	39
Annexe 1 - Théorie des réseaux de neurones [15-19] .....	39
Les réseaux de neurones artificiels .....	39
Réseau à propagation directe (feedforward) .....	39
Perceptron multicouche .....	39
Couche de convolution .....	41
Couche de pooling .....	42
Réseaux de neurones convolutifs .....	42
Apprentissage .....	43
Algorithme du gradient .....	43
Rétropropagation .....	43
L'algorithme du gradient stochastique .....	44
Optimiseurs .....	44
Surapprentissage .....	45
Évaluation du réseau .....	45
Hyperparamètres .....	46
Le réglage des hyperparamètres et l'ensemble de validation .....	46
Annexe 2 - Évolution des mesures du modèle .....	47
Bibliographie .....	50

## Table des Figures

Fig 1 - (à gauche) Indication des plans anatomiques standard, (à droite) CT scan thoracique rendu dans les différents plans anatomiques .....	8
Fig 2 - Exemple d'un ensemble de coupes transversales du cerveau obtenues à partir d'un CT scan .....	9
Fig 3 - Quelques images de l'ensemble des données .....	11
Fig 4 - Schéma de l'organisation des répertoires (En gris sont les répertoires et en blanc les fichiers) .....	12
Fig 5 - Graphique des tailles des images .....	12
Fig 6 - Ajustement linéaire des moindres carrés aux dimensions des images .....	13
Fig 7 - Exemple d'images de l'ensemble de données avec des bords blancs.....	15
Fig 8 - Comparaison de CT (à gauche) et de CTA (à droite) .....	16
Fig 9 - Quelques images avec des annotations.....	16
Fig 10 - Deux types de distorsion des couleurs. Images réelles (à gauche) et variance entre les canaux (à droite). Les annotations en couleur (en haut), la distorsion intrinsèque de couleur (en bas).....	17
Fig 11- Sous-domaines du machine learning et quelques applications .....	20
Fig 12 - Validation croisée à k plis.....	22
Fig 13 - Fichier où sont enregistrées les configurations des modèles.....	24
Fig 14 - Informations affichées pendant l'apprentissage.....	26
Fig 15 - Illustration d'une matrice de confusion .....	26
Fig 16 - Architecture VGG16 .....	30
Fig 17 - DenseNet.....	34
Fig 18 - Architectures DenseNet .....	34
Fig 19 - Grad-cam pour les trois modèles testés .....	35
Fig 20 - Perceptron .....	40
Fig 21 - Perceptron multicouche (chaque nœud est un perceptron) .....	40
Fig 22 - Opération de convolution .....	41
Fig 23 - Architecture standard de réseau neuronal convolutif.....	42
Fig 24 - Exemple de graphique de perte d'apprentissage et de test.....	46
Fig 25 - Évolution des mesures du VGG16 .....	47
Fig 26 - Évolution des mesures du ResNet-50 .....	48
Fig 27 - Évolution des mesures du DenseNet-169 .....	49

## Contexte

Ce travail a été réalisé en 2 mois dans le cadre du stage 2A. Le stage a été exécuté à l'ENSTA Bretagne dans l'équipe STIC-REMS et encadré par monsieur Jean-Christophe Cexus et monsieur Abdelmalek Toumi.

## Introduction

La maladie à coronavirus 2019 (COVID-19) est une maladie infectieuse causée par le coronavirus 2 du syndrome respiratoire aigu sévère (SRAS-CoV-2). Elle a été identifiée pour la première fois en décembre 2019 à Wuhan, en Chine. Les complications peuvent comprendre la pneumonie, le syndrome de détresse respiratoire aiguë (SDRA), la défaillance de plusieurs organes, le choc septique et le décès. Les complications cardiovasculaires peuvent comprendre l'insuffisance cardiaque, les arythmies, l'inflammation cardiaque et les caillots sanguins. Au 5 septembre 2020, plus de 26,6 millions de cas ont été signalés dans 188 pays et territoires, avec plus de 874 000 décès [9]. Outre les effets directs sur la santé, le déclenchement de la pandémie a d'autres effets indirects, tels qu'une diminution de la disponibilité des hôpitaux pour le traitement d'autres maladies, une baisse de l'activité économique, la fermeture d'écoles, etc.

Comme il s'agit d'un virus relativement nouveau, il n'existe actuellement aucun traitement ou remède officiel, de sorte que la meilleure façon de lutter contre la maladie est d'empêcher sa propagation. Pour y parvenir, les gouvernements du monde entier recommandent différentes mesures de sécurité, telles que l'utilisation de masques, de désinfectants pour les mains et la distanciation sociale. Quoi qu'il en soit, pour pouvoir faire face efficacement à la situation, il est crucial de pouvoir identifier les personnes infectées, afin qu'elles puissent recevoir les soins médicaux appropriés et, en fin de compte, établir une quarantaine et limiter la propagation du virus.

Il existe actuellement deux catégories de tests pour diagnostiquer une infection courante [1] : les tests moléculaires, qui détectent le matériel génétique du virus, et les tests antigènes qui détectent des protéines spécifiques à la surface du virus. Avec les tests moléculaires, les résultats sont obtenus en un jour environ et sont généralement fiables. En revanche, les tests antigéniques permettent d'obtenir des résultats en moins d'une heure, mais ces résultats sont moins fiables. Actuellement, le test le plus utilisé est un test moléculaire connu sous le nom d'amplification en chaîne par polymérase à transcription inverse (RT-PCR). Pour effectuer ce test, un échantillon est prélevé à partir de prélèvements nasaux ou de gorge et est analysé par des techniciens qualifiés. Au début de la pandémie, en particulier, on a constaté une pénurie de tests dans différentes régions du monde ainsi qu'un délai plus long pour obtenir les résultats.

Alors que les tests rapides de RT-PCR deviennent de plus en plus disponibles, des défis demeurent, notamment des taux élevés de faux négatifs, des variabilités dans les techniques de test et une sensibilité parfois rapportée aussi faible que 60-70% [2]. En conséquence, les médecins utilisent souvent des informations complémentaires, telles que les CT scans pour parvenir à un diagnostic final. Un CT scan, ou tomodensitométrie, est une procédure d'imagerie médicale qui utilise des combinaisons traitées par ordinateur de nombreuses mesures de rayons



X prises sous différents angles pour produire des images en coupe transversale, permettant à l'utilisateur de voir à l'intérieur de l'objet sans le couper. Les CT scans de la poitrine sont utilisés régulièrement et de manière fiable par les professionnels de la santé pour aider à diagnostiquer les causes des signes cliniques ou des symptômes de maladie de la poitrine, comme la toux, l'essoufflement, les douleurs thoraciques ou la fièvre. Les CT scans de la poitrine peuvent mettre en évidence divers troubles pulmonaires, tels que : tumeurs bénignes et malignes, pneumonie, tuberculose, etc. [3]

En effet, la COVID-19 se manifeste dans les CT scans du thorax par différentes anomalies visibles [4,5] et une étude montre une corrélation significative entre les résultats des tests RT-PCR et le diagnostic par CT scans du thorax effectué par des professionnels qualifiés [6], ouvrant la possibilité de détection par l'analyse de ces images. Néanmoins, il n'est actuellement pas conseillé de n'utiliser que les CT scans thoraciques pour diagnostiquer la maladie en raison du manque d'informations concluantes sur la précision de cette approche [7]. Une déclaration de consensus multinationale de la Fleischner Society propose les lignes directrices suivantes pour l'utilisation des images médicales dans la gestion des patients pendant la pandémie de COVID-19 :

- L'imagerie n'est pas indiquée chez les patients soupçonnés d'être atteints de la maladie à coronavirus 2019 (COVID-19) et présentant des caractéristiques cliniques bénignes, à moins qu'ils ne soient exposés à un risque de progression de la maladie.
- L'imagerie est indiquée chez un patient atteint de la COVID-19 et dont l'état respiratoire s'aggrave.
- Dans un environnement aux ressources limitées, l'imagerie est indiquée pour le triage médical des patients soupçonnés d'être atteints de la COVID-19 qui présentent des caractéristiques cliniques modérées à graves et une forte probabilité de maladie avant le test.

Le présent travail étudie l'utilisation de l'intelligence artificielle, et plus particulièrement le deep learning, pour créer un algorithme capable d'analyser automatiquement les CT scans de la poitrine et de produire un diagnostic de la COVID-19. L'algorithme pourrait être utilisé dans les situations décrites précédemment, pour compléter les résultats d'autres tests, et éventuellement dans d'autres scénarios, à mesure que de nouvelles informations sur la validité de l'utilisation des CT scans thoraciques dans la gestion des patients pendant la pandémie de COVID-19 seront disponibles. Il pourrait remplacer l'analyse traditionnelle par CT scan par un professionnel de la santé comme une approche plus rapide et plus précise ou compléter le travail du professionnel de la santé.

## **1) Analyse des données**

### **1.2) CT scan (tomographie assistée par ordinateur)**

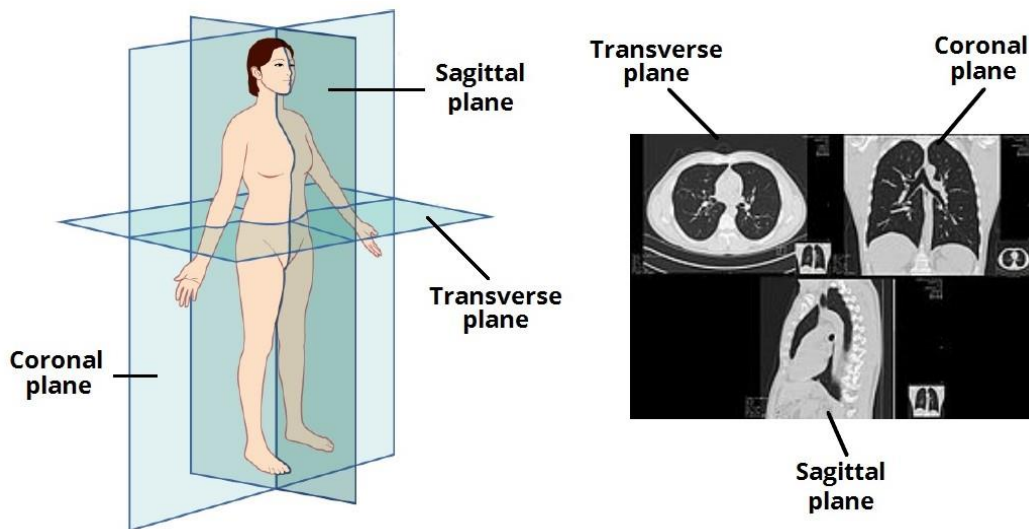
Tout comme la radiographie conventionnelle, dans les CT scans, les rayons X sont émis vers le patient et différents matériaux à l'intérieur du corps vont absorber les rayons X dans des

quantités différentes. Les rayons X sont ensuite analysés après avoir traversé le corps et, en fonction de l'atténuation, l'image est créée. Comme ce qui est représenté par les images radiographiques est l'intensité de l'atténuation des rayons X, les images radiographiques sont généralement affichées en niveaux de gris. Normalement, les matériaux plus denses ont tendance à absorber plus de rayons X et les matériaux moins denses en absorbent moins. Ce matériau plus dense apparaîtra souvent clairement tandis que le matériau moins dense sera plus foncé sur l'image finale.

Pour effectuer un CT scan, le patient est introduit en position couchée dans l'appareil, où un faisceau étroit de rayons X est dirigé sur le patient et tourné rapidement autour du corps pour fournir une série d'images sous différents angles. Le résultat de cette procédure est un volume de voxels, qui sont des pixels tridimensionnels. Ceux-ci peuvent être présentés à un observateur humain par différentes méthodes :

- Tranche fine : On considère généralement qu'il s'agit de plans représentant une épaisseur inférieure à 3 mm
- Projection, y compris la projection d'intensité maximale et la projection d'intensité moyenne
- Représentation en volume (VR).

L'un des avantages du CT scan est que les données collectées peuvent être utilisées pour créer un modèle tridimensionnel de l'objet inspecté, une procédure connue sous le nom de rendering de volume. Même si l'on souhaite obtenir des images en deux dimensions, celles-ci peuvent être créées sous différents angles. Les plans de visualisation standard en anatomie sont les suivants :

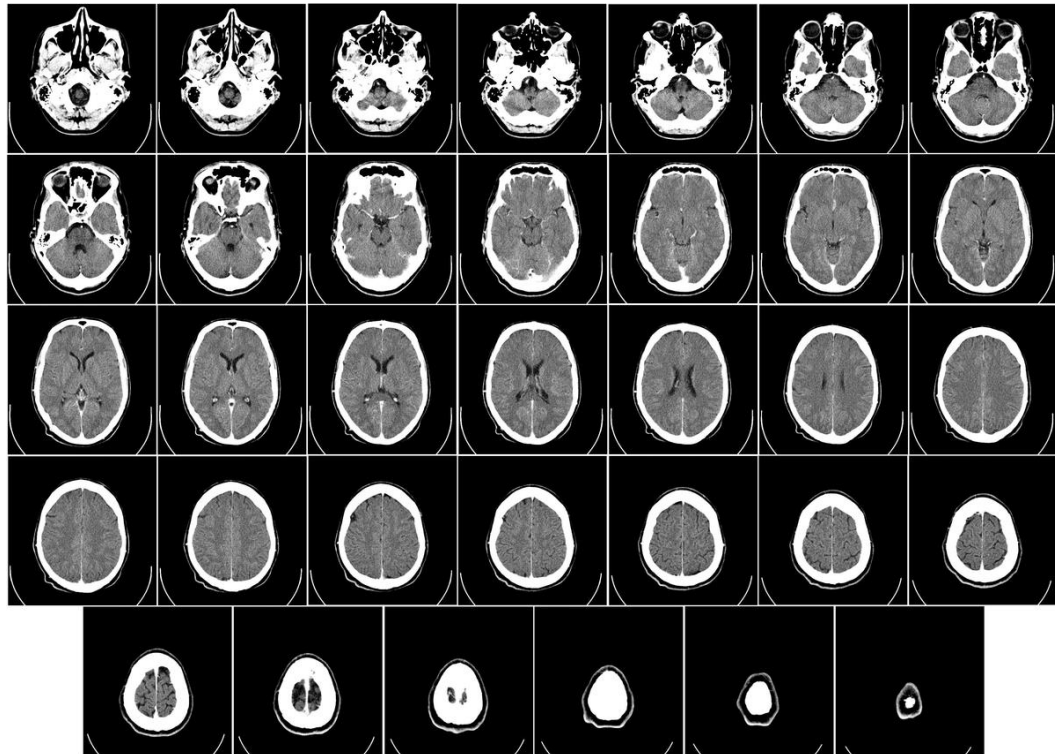


*Fig 1 - (à gauche) Indication des plans anatomiques standard, (à droite) CT scan thoracique rendu dans les différents plans anatomiques*

Si l'on choisit de visualiser les données sous forme de tranches fines, comme c'est le cas dans cette étude, le résultat sera un ensemble de multiples images bidimensionnelles en niveaux



de gris, dans lequel chaque image est une tranche de l'objet légèrement éloignée de la tranche précédente.



*Fig 2 - Exemple d'un ensemble de coupes transversales du cerveau obtenues à partir d'un CT scan*

### 1.3) L'ensemble des données

L'un des principaux obstacles à l'expérimentation de l'utilisation du deep learning pour le diagnostic de la COVID-19 et d'applications similaires est la rareté des données disponibles. Le deep learning a tendance à être une tâche très exigeante en matière de données et ce manque de données avec lesquelles travailler pose un sérieux problème. En fait, ces données existent et sont suffisamment nombreuses pour être utiles pour le deep learning, mais elles ne sont le plus souvent pas accessibles au public, comme c'est normalement le cas des images médicales, pour des raisons de confidentialité des patients. Malgré cela, au fil du temps, de plus en plus de ces données deviennent accessibles au public. Le problème avec la COVID-19 est qu'il s'agit d'une maladie si récente que les données disponibles au public sont encore insuffisantes pour une application réussie du deep learning.

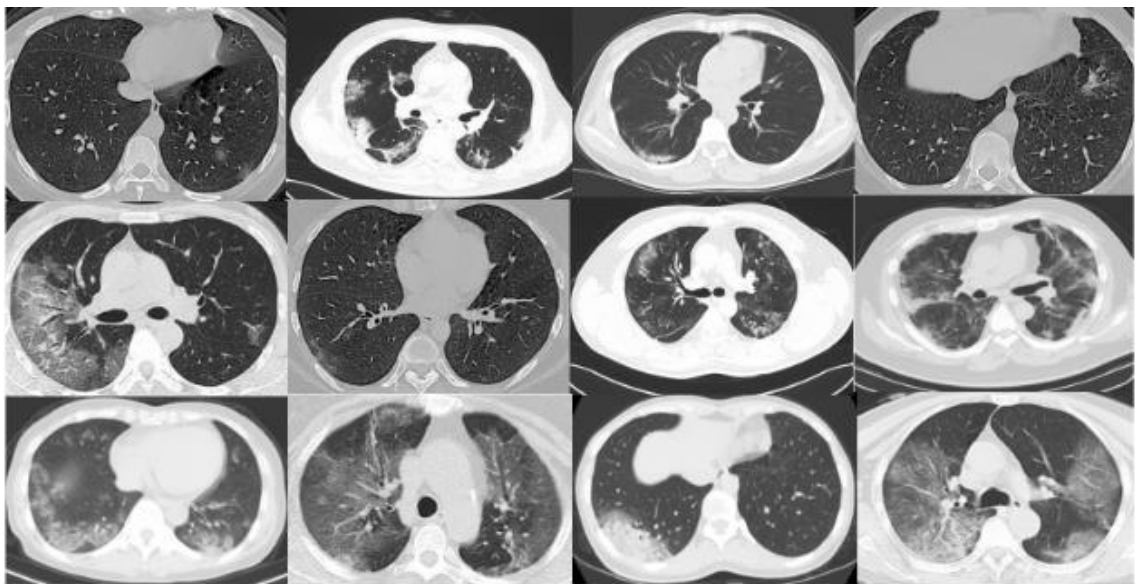
Dans le but de surmonter ce problème, Zhao et al. [8] ont proposé une approche pour créer une base de données des coupes transversales du thorax des patients atteints de la COVID-19 et ont affirmé que cette méthode leur permettait de créer la plus grande base de données publique de CT scans COVID-19 disponible à l'époque. La méthode consiste essentiellement à extraire les CT scans thoraciques des articles scientifiques concernant la

COVID-19 qui incluent des CT scans thoraciques de patients atteints de la COVID-19 à titre d'illustration.

Afin de créer l'ensemble de données, 760 prépublications sur la COVID-19 de medRxiv et bioRxiv postées du 19 janvier au 25 mars ont été collectées. Ces sites web distribuent gratuitement des prépublications, c'est-à-dire des articles non publiés qui n'ont pas encore été soumis à un examen par les pairs, dans le domaine médical. Beaucoup de ces prépublications présentent des cas de COVID-19 et certaines d'entre elles contiennent des CT scans de la poitrine. Ces figures sont souvent accompagnées de légendes, à partir desquelles on peut trouver des informations utiles, par exemple si le patient a été diagnostiqué comme étant atteint de la COVID-19. Pour faciliter l'extraction des informations pertinentes des fichiers PDF, PyMuPDF a été utilisé. Il s'agit d'une bibliothèque python utilisée pour la manipulation des PDF qui a été utile pour extraire les informations de structure de bas niveau et localiser toutes les figures intégrées. A partir des informations de structure, les légendes associées aux figures ont également été identifiées. Compte tenu de ces figures et légendes extraites, tous les CT scans ont été sélectionnés manuellement. Ensuite, pour chaque CT scan, la légende associée a été lue afin de juger si elle est positive pour la COVID-19. Si la légende ne permettait pas de juger, le texte analysant cette figure dans le document était localisé pour prendre une décision. En fin de compte, l'ensemble de données contient 349 CT scans positifs de COVID-19 provenant de 213 patients. Pour compléter l'ensemble de données, des CT scans de la poitrine de patients ne présentant pas de COVID-19 ont également été collectés. Ces images ont été collectées directement à partir des bases de données suivantes : La base de données MedPix, l'ensemble de données LUNA, le site web Radiopaedia et PubMed Central. Au total, 397 CT scans de la poitrine de 171 patients ne présentant pas de COVID-19 ont été collectés. Le tableau ci-dessous résume la composition de l'ensemble de données.

	Nombre d'images	Nombre de patients
<b>COVID</b>	349	213
<b>Non COVID</b>	397	171

*Tableau 1 - Résumé de la composition de l'ensemble de données*



*Fig 3 - Quelques images de l'ensemble des données*

Il est très important de comprendre que, comme indiqué précédemment, un CT scan, s'il est choisi d'être visualisé en tranches, est un ensemble d'images. Cependant, dans cet ensemble de données, il n'y a souvent que quelques images par patient et dans certains cas une seule, ce qui signifie que l'ensemble complet n'est pas disponible et qu'il manque des images. Cela signifie que pour certains patients, nous pouvons avoir une tranche prise à une hauteur particulière et pour un autre patient une tranche à une hauteur différente. Malheureusement, cette information sur la tranche à laquelle correspond l'image n'est pas disponible. Il est également important de reconnaître que l'ensemble de données peut contenir plusieurs images (différentes tranches) d'un même patient. Cela sera pertinent plus tard, lors de la division de l'ensemble de données en sous-ensembles, puisque ces images provenant du même patient ont tendance à être fortement corrélées.

L'ensemble de données est organisé de la manière suivante : il y a un répertoire appelé CT\_COVID, où sont stockées toutes les images des patients atteints de la COVID-19 et il y a un répertoire appelé CT\_NonCOVID où sont stockées toutes les images des patients qui n'ont pas la COVID-19. En dehors de ce répertoire, il y a un répertoire appelé info et à l'intérieur de ce répertoire, il y a deux fichiers, COVID-CT-MetaInfo.csv et NonCOVID-CT-MetaInfo. Ces fichiers indiquent à quel patient appartient chaque image.

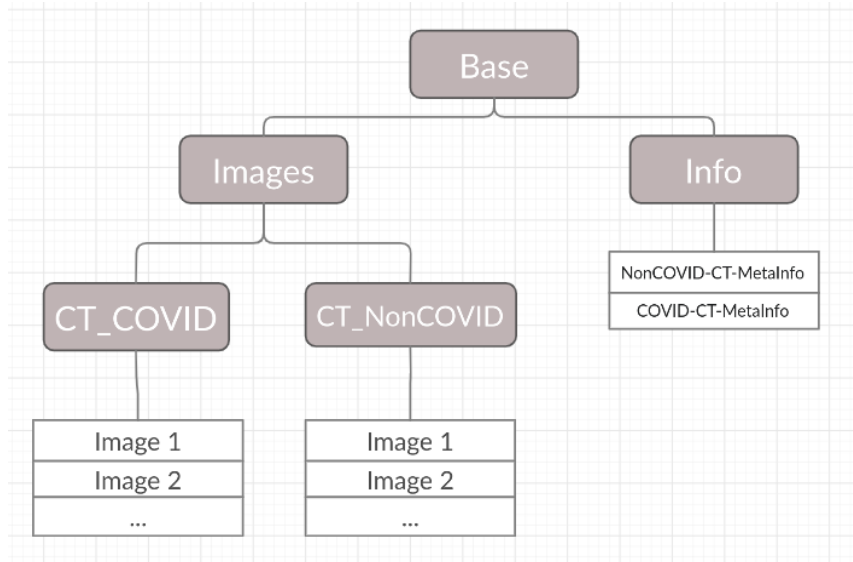


Fig 4 - Schéma de l'organisation des répertoires (En gris sont les répertoires et en blanc les fichiers)

Comme les images de l'ensemble de données sont tirées de différents articles scientifiques, on s'attend à ce que l'ensemble présente une certaine hétérogénéité en ce qui concerne les caractéristiques des images. Tout d'abord, ces images ont des tailles différentes. Le graphique ci-dessous montre un tracé de toutes les tailles d'images :

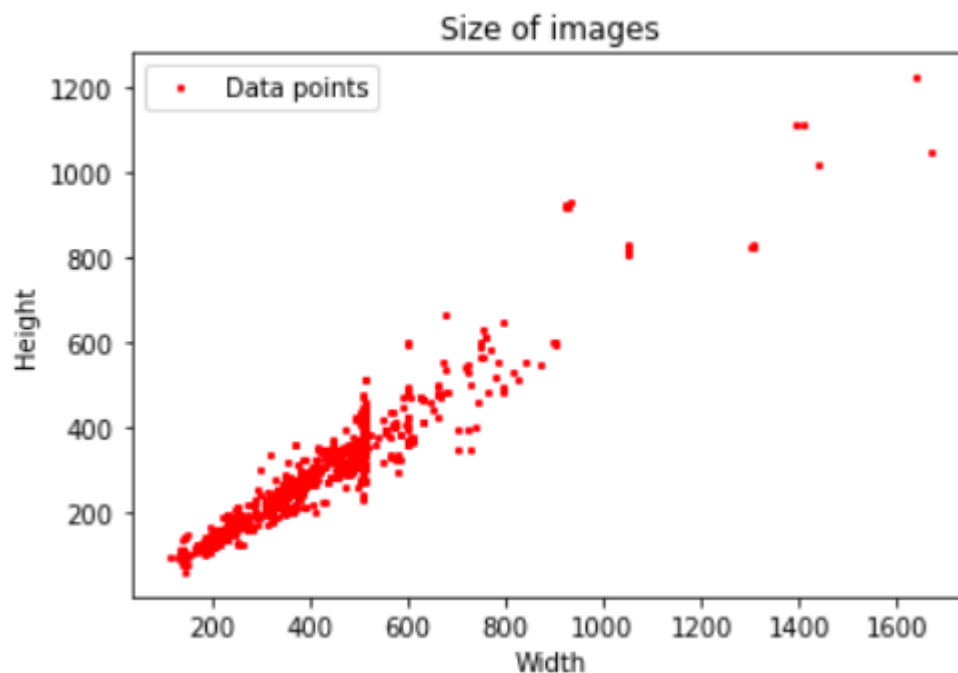


Fig 5 - Graphique des tailles des images

Le tableau ci-dessous résume les statistiques des tailles des images :

	Minimum	Maximum	Moyenne	Médiane	Écart-type
<b>Largeur (pixels)</b>	115	1671	425,48	407	188,42
<b>Hauteur (pixels)</b>	61	1225	302,55	291	145,11

Tableau 2 - Statistiques sur la taille des images

En analysant le graphique précédent, nous pouvons observer que les largeurs et les hauteurs ont une variance élevée. Cependant, les points de données semblent présenter une relation linéaire, ce qui implique que les rapports largeur/hauteur ont une faible variance. En d'autres termes, même si les images peuvent avoir des tailles significativement différentes, elles semblent avoir plus ou moins le même rapport largeur/hauteur. Le graphique suivant montre une ligne de régression des moindres carrés ajustée aux points de données.

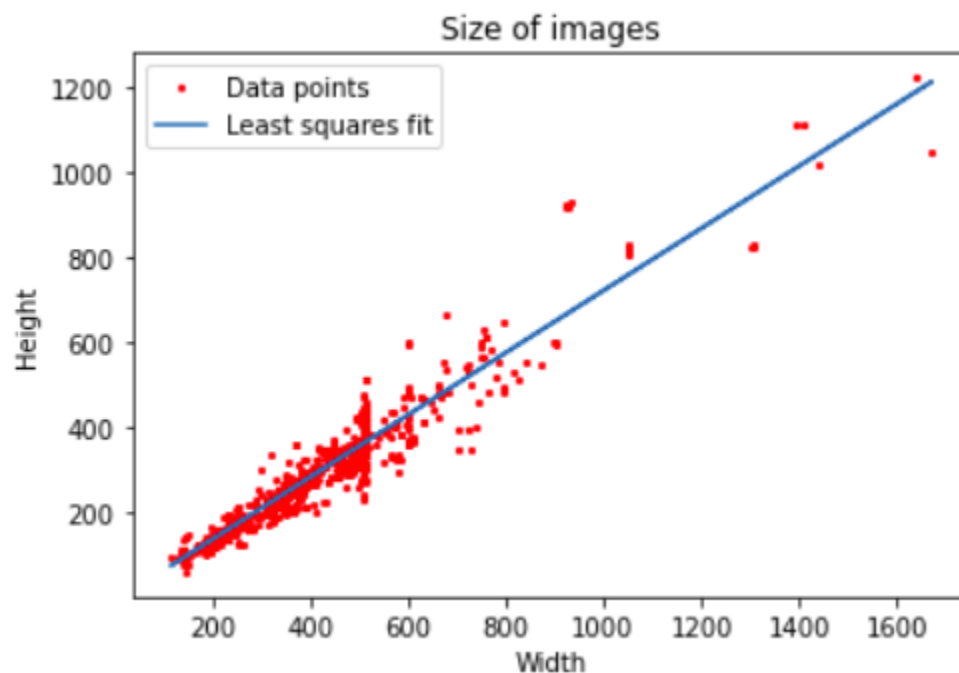


Fig 6 - Ajustement linéaire des moindres carrés aux dimensions des images

Le tableau ci-dessous résume les statistiques des rapports largeur/hauteur des images :

	Minimum	Maximum	Moyenne	Médiane	Moindres carrés	Écart-type
<b>Rapport d'aspect (largeur/hauteur)</b>	0,94	2,43	1,44	1,42	1,36	0,19

Tableau 3 - Statistiques des rapports d'aspect

En ce qui concerne le format des images, il y a à la fois des images JPEG et PNG dans l'ensemble, le tableau suivant résume la répartition :

	JPEG	PNG
<b>COVID</b>	34	315
<b>Non COVID</b>	202	195
<b>Total</b>	236	510

Tableau 4 - Distribution des formats de fichiers

Les images ont également différents modes, le tableau suivant résume la répartition des modes :

	Niveaux de gris	RGB	RGBA
<b>COVID</b>	21	154	174
<b>Non COVID</b>	23	374	0
<b>Total</b>	44	528	174

Tableau 5 - Répartition des modes dans l'ensemble de données

En analysant le canal alpha des images RGBA, il est possible de découvrir qu'il ne contient aucune information. Cela est vrai pour toutes les images RGBA et elles peuvent donc être converties en RGB sans perte d'information.

Le mode RGB peut également être utilisé pour représenter des images en niveaux de gris, ce qui se produit lorsque les trois canaux sont identiques. Par conséquent, afin de découvrir le nombre réel d'images en couleur, il faut inspecter la variance entre les canaux. S'il n'y a pas de variance, alors l'image est juste une image en niveaux de gris représentée en RGB. En revanche, s'il y a une variance, alors il s'agit d'une vraie image en couleur. En analysant la variance des canaux pour toutes les images RGB, nous obtenons les résultats suivants :

	Vrai niveau de gris	Vraie couleur
<b>COVID</b>	248	101
<b>Non COVID</b>	350	47
<b>Total</b>	598	148

Tableau 6 - Distribution des vraies images en niveaux de gris et des vraies images en couleurs

À ce stade, il est important de se rappeler que les CT scans sont censés être en niveaux de gris, comme mentionné dans la section précédente sur les CT scans. Cependant, parmi nos images, nous avons des images en couleur et cela signifie qu'il s'agissait autrefois d'images en niveaux de gris qui ont subi une certaine distorsion. La section suivante analyse la source de ces distorsions.



## 1.4) Pollution des données

Étant donné l'origine des images, il n'est pas surprenant qu'elles puissent présenter certaines distorsions et artefacts. Cette section analyse cette pollution des données.

### 1.4.1) Bords blancs

Certaines images ont des bords blancs. Ces informations sont inutiles pour le diagnostic de la COVID-19 et peuvent être mal utilisées par l'algorithme de deep learning.

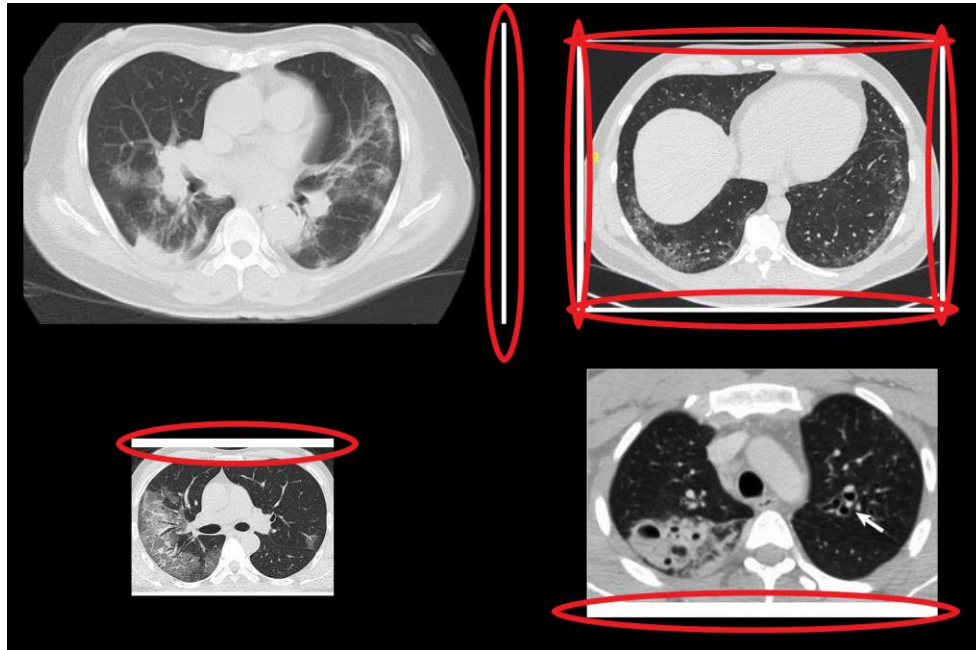
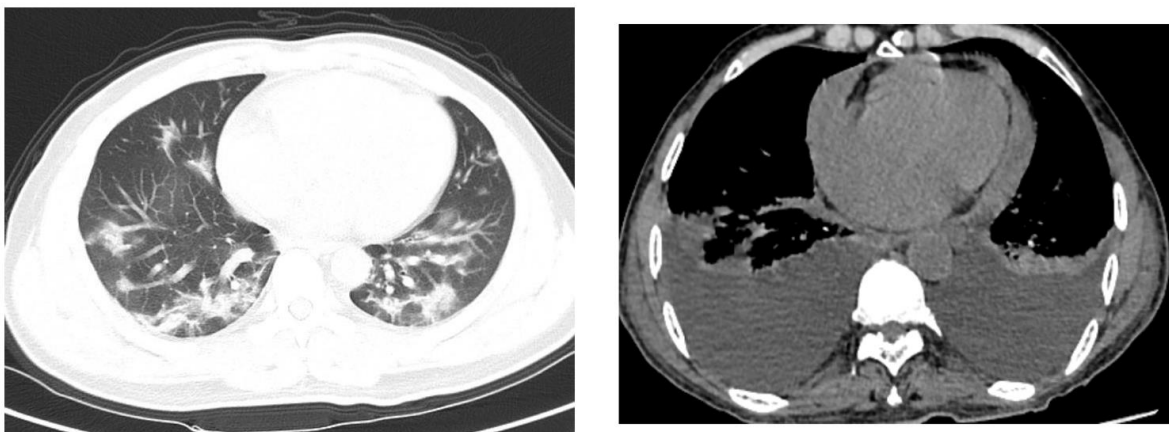


Fig 7 - Exemple d'images de l'ensemble de données avec des bords blancs

### 1.4.2) Inclusion incorrecte de l'angiographie par tomодensitométrie (CTA)

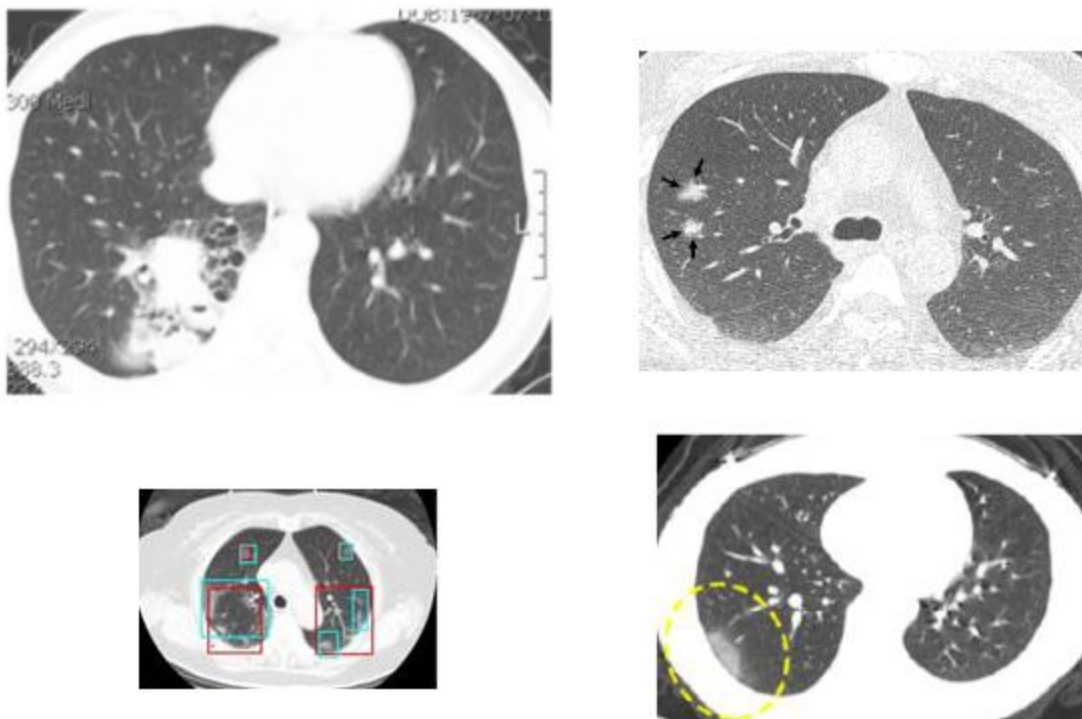
L'angiographie par tomодensitométrie est une technique de tomographie assistée par ordinateur utilisée pour visualiser les vaisseaux artériels et veineux dans tout le corps. Pour ce faire, une substance est injectée dans les vaisseaux sanguins, afin de créer un contraste, avant d'entrer dans l'appareil de tomодensitométrie. Les CTA scans ont un aspect sensiblement différent des CT scans ordinaires et quelques-uns ont été inclus dans cet ensemble de données. Ces CTA scan constituent un type de données différent, et leur inclusion dans l'ensemble de données pourrait conduire à l'apprentissage de modèles qui sont contre-productifs pour les CT scans.



*Fig 8 - Comparaison de CT (à gauche) et de CTA (à droite)*

### 1.4.3) Annotations

Comme les images proviennent d'articles scientifiques qui les utilisent comme illustration, certaines d'entre elles comportent des annotations, telles que du texte, des formes, des flèches, etc. qui se superposent à l'image. Ces annotations ajoutent des informations inutiles qui pourraient être mal utilisées par l'algorithme de deep learning et bloquent également certaines informations qui pourraient être utiles.

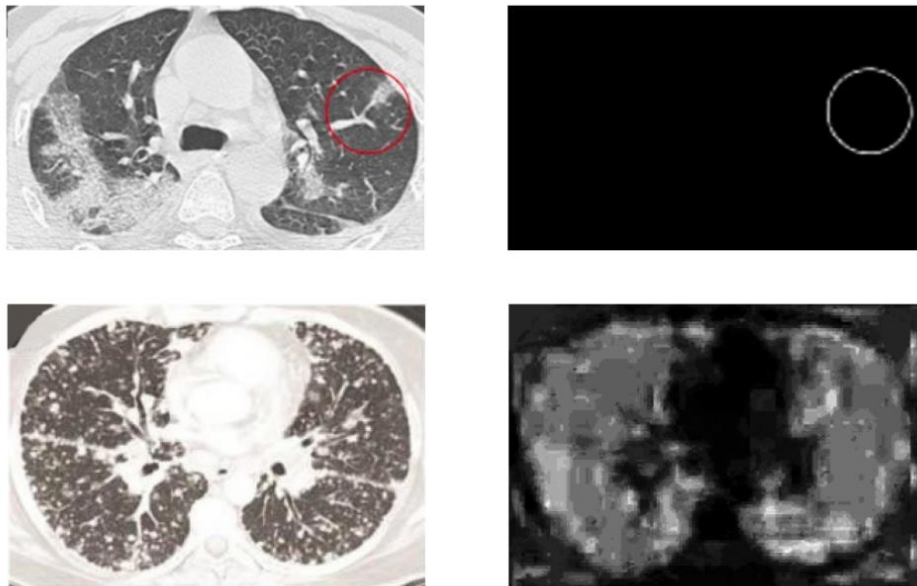


*Fig 9 - Quelques images avec des annotations*

On peut observer que certaines annotations sont en niveaux de gris et d'autres en couleur.

#### 1.4.4) Distorsion des couleurs

Comme indiqué précédemment, certaines images sont en couleur alors qu'elles devraient être en niveaux de gris. Il existe deux principaux types de distorsion de couleur parmi ces images : l'introduction d'annotations en couleur et ce que j'appellerai "la distorsion intrinsèque de couleur", dans laquelle les pixels qui représentent l'image réelle ont une certaine couleur. Il est pertinent de noter que pour la plupart des images qui ont une distorsion intrinsèque de couleur, il est très difficile de remarquer cette couleur. La figure suivante montre des exemples de ces deux types de distorsion des couleurs.



*Fig 10 - Deux types de distorsion des couleurs. Images réelles (à gauche) et variance entre les canaux (à droite). Les annotations en couleur (en haut), la distorsion intrinsèque de couleur (en bas)*

Les images de droite dans la figure précédente montrent la variance entre les canaux pour chaque pixel et permettent de visualiser où la couleur existe réellement. Les pixels qui ont réellement de la couleur sont ceux qui apparaissent en clair sur ces images. On peut remarquer que pour l'image du haut, la seule couleur existante est celle introduite par l'annotation alors que dans l'image du bas, on voit qu'il y a de la couleur partout dans l'image originale.

## **2) Deep learning**

### **2.1) Ressources**

#### **2.1.1) Matériel**

Au début, j'ai essayé d'utiliser mon ordinateur personnel, équipé d'un processeur Intel i3 quad core 2,4 GHz et de 4 GB de RAM, pour entraîner les modèles de deep learning. J'ai vite découvert que cet ordinateur n'était pas assez puissant et qu'il manquait souvent de mémoire ou que les calculs étaient trop longs à effectuer. Afin de résoudre ce problème, j'ai commencé à utiliser google colab. Google colab est un service gratuit qui permet d'accéder aux CPU et GPU des serveurs google. Ces ressources sont partagées entre tous les utilisateurs, il n'y a donc aucune garantie de disponibilité et il y a également certaines limitations concernant le temps d'utilisation, en particulier pour les GPU. Ainsi, si vous passez trop de temps à utiliser les GPU, votre accès aux GPU sera bloqué pendant un certain temps. Plusieurs CPU et GPU différents sont disponibles dans google colab, bien qu'il ne soit pas possible de choisir lequel utiliser, il vous est attribué automatiquement. Parmi les GPU disponibles, il y a les Nvidia K80, T4, P4 et P100. L'interface de google colab est très similaire à celle de Jupyter Notebook, qui est standard dans les expériences de machine learning. Les CPU disponibles dans google colab ne présentaient pas une amélioration très significative par rapport au CPU de mon ordinateur, mais, les GPU ont augmenté sensiblement le temps d'exécution et m'ont permis d'expérimenter avec les architectures de deep learning les plus populaires. Finalement, j'ai rencontré des problèmes concernant la disponibilité des GPU dans google colab et j'ai donc cherché des services similaires pour le remplacer. L'une des possibilités était Microsoft Azure. Cependant, ils n'offrent que l'accès aux CPU gratuitement et pour pouvoir accéder aux GPU, il faut payer. Comme leurs CPU n'ont pas donné de bons résultats dans mes expériences, j'ai continué à utiliser google colab pour l'ensemble de mon travail.

#### **2.1.2) Logiciels**

Tout le code a été développé en Python et PyTorch a été utilisé comme bibliothèque Python deep learning. PyTorch a été choisi parce que l'article sur lequel ce travail est basé fournit des exemples en PyTorch [8]. Le code est organisé dans un fichier Jupyter Notebook puisque c'est l'entrée acceptée par google colab.

#### **2.1.3) Github**

L'ensemble du code produit est disponible dans le répertoire github suivant : <https://github.com/tonydosreis/deeplearning-covid19>

## 2.2) Définition du problème comme un problème de machine learning

Notre principal objectif est de produire une fonction qui reçoit en entrée une tranche de CT scan thoracique transversale et produit en sortie un diagnostic : COVID ou non COVID. En d'autres termes, l'entrée est une image et la sortie est un label sur deux possibles. Pour ce faire, nous disposons de données provenant de patients COVID-19 et d'autres données provenant de patients qui n'ont pas de COVID-19 et nous espérons apprendre à partir de ces données comment faire la distinction entre les deux et diagnostiquer la COVID-19. Cet objectif d'apprentissage automatique à partir des données constitue un problème de machine learning.

L'ensemble de données est organisé de telle sorte que les images des patients diagnostiqués avec la COVID-19 se trouvent dans un dossier et les images des patients non diagnostiqués avec la COVID-19 se trouvent dans un autre dossier. Cela signifie que pour chaque image de l'ensemble de données, nous connaissons déjà le diagnostic correct grâce au dossier auquel l'image appartient. En d'autres termes, pour toutes les images de l'ensemble de données, nous connaissons déjà la sortie correcte de cette fonction de diagnostic que nous recherchons. Le fait que nous disposions d'un échantillon d'entrées et des sorties correspondantes signifie que nous sommes dans un scénario d'apprentissage supervisé, qui est un sous-domaine du machine learning.

En outre, la nature de la sortie est discrète : une catégorie parmi deux possibilités. Cela implique que notre problème est un problème de classification. De plus, comme il n'y a que deux labels ou classes, il s'agit d'un problème de classification binaire. Ensuite, comme les classes sont mutuellement exclusives, vous ne pouvez pas être diagnostiqué avec la COVID-19 et en même temps ne pas être diagnostiqué avec la COVID-19, le problème est un problème de label unique. Enfin, le nom complet du problème est un problème d'apprentissage supervisé de classification binaire à label unique.

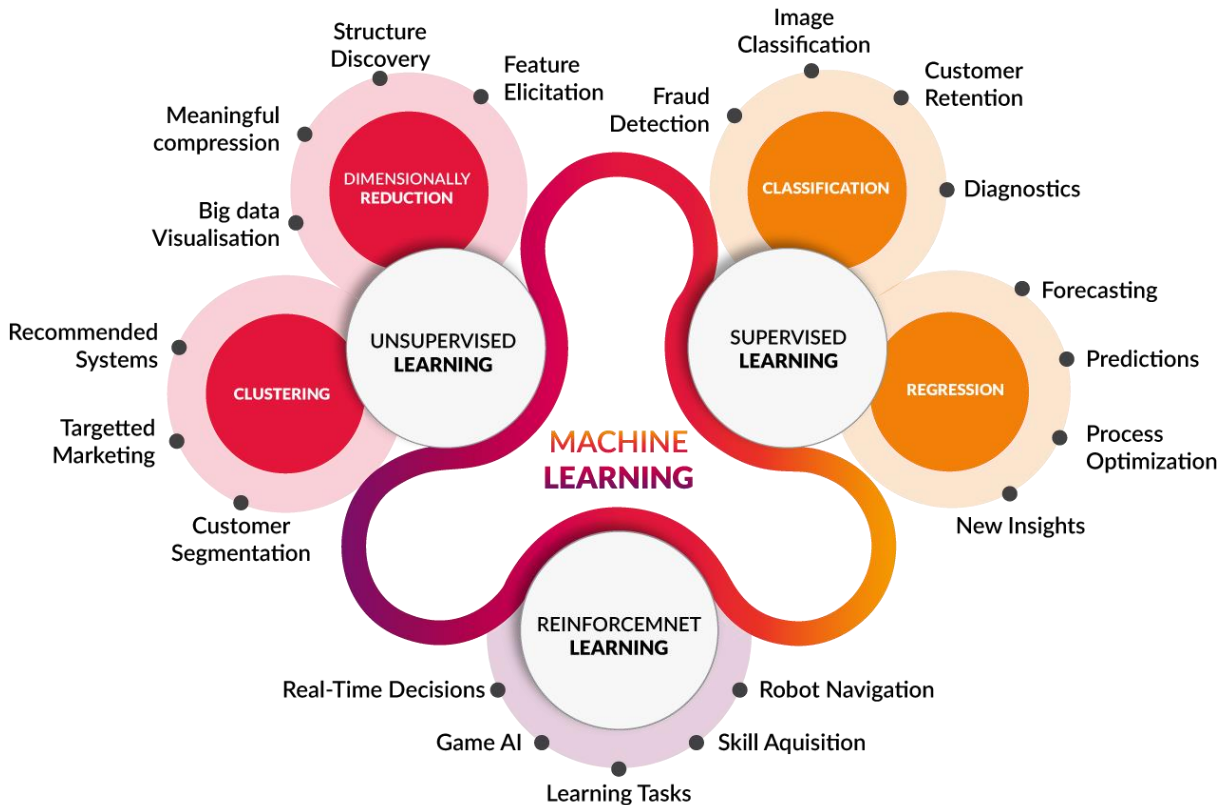


Fig 11- Sous-domaines du machine learning et quelques applications

Il existe plusieurs approches pour traiter ce problème. Historiquement, l'approche la plus efficace lorsque les entrées sont des images est l'utilisation de réseaux neuronaux convolutifs, car ils apprennent hiérarchiquement des représentations plus abstraites et sont insensibles aux translations, ce qui est quelque peu similaire à la façon dont les humains apprennent à partir d'images. Les réseaux neuronaux convolutifs sont juste un type de réseau neuronal qui comprend des opérations de convolution, qui sont utiles dans les tâches d'identification visuelle. Chaque fois qu'un réseau neuronal est utilisé pour résoudre un problème du machine learning, on parle de deep learning.

L'annexe 1 contient un simple résumé sur les réseaux neuronaux qui définissent tous les termes techniques utilisés dans le reste de ce document.

### 2.3) Préparation des données

Avant d'apprendre à partir des données, il y a des traitements préalables qui doivent être effectués sur les images et d'autres qui sont facultatifs, mais qui pourraient faciliter le processus d'apprentissage.



### **2.3.1) Exclusion des images CTA**

Les images CTA incorrectement incluses ont simplement été supprimées de l'ensemble de données.

### **2.3.2) Conversion en niveaux de gris**

Pour traiter la distorsion des couleurs, les images ont été converties en niveaux de gris. La distorsion des couleurs causée par l'introduction d'annotations en couleur n'est pas un problème parce que les annotations sont colorées, mais parce que les annotations bloquent une partie de l'image. La conversion de ces images en niveaux de gris ne résout pas ce problème, mais ne l'aggrave pas non plus. Le second type de distorsion des couleurs, que j'ai appelé "distorsion intrinsèque des couleurs" n'est apparemment pas très fort, donc en convertissant ces images en niveaux de gris, elles ne changent pas de manière significative.

### **2.3.3) Normalisation**

La normalisation est une technique de prétraitement très courante. Normalement, les canaux de l'image sont normalisés de telle sorte que la moyenne soit égale à zéro et la variance à un. Dans ce travail, une technique de normalisation différente qui sera expliquée plus tard a été utilisée.

## **2.4) Validation croisée à k plis**

En particulier pour les petits ensembles de données qui n'ont pas suffisamment d'échantillons représentant bien les principales caractéristiques de la population, on peut observer une variation sur la performance d'un modèle mesuré sur l'ensemble de validation selon la façon dont les données ont été divisées en ensembles d'apprentissage et de validation. En d'autres termes, lorsque certains éléments appartiennent à l'ensemble d'apprentissage tandis que d'autres appartiennent à l'ensemble de validation, on observera une certaine performance. Mais si nous mélangeons les données de manière aléatoire, en créant de nouveaux ensembles d'apprentissage et de validation et en relançant le processus depuis le début, nous observerons une performance différente sur l'ensemble de validation. En effet, comme l'ensemble de données est petit, il peut y avoir des images dans l'ensemble de validation qui contiennent des caractéristiques importantes à apprendre et, en même temps, il peut ne pas y avoir d'images similaires dans l'ensemble d'apprentissage. Ceci est problématique, car chaque fois que nous effectuons l'apprentissage, nous obtenons des résultats différents. Une solution à ce problème est la validation croisée à k plis.

Dans la validation croisée à k plis, l'ensemble de données est divisé en k partitions égales et k itérations sont exécutées pour chaque époque. Dans chaque itération, une partition est l'ensemble de validation et les autres sont l'ensemble d'apprentissage. Les mesures sont collectées à chaque itération et à la fin de toutes les itérations, la moyenne des mesures est calculée pour obtenir les mesures de l'époque.



Fig 12 - Validation croisée à k plis

Dans cette approche, vous évitez la variance qui provient des différentes façons de diviser les données en incorporant ces différences dans une mesure moyenne.

### 2.4.1) Division de l'ensemble des données

Comme l'ensemble de données est relativement petit, j'ai choisi d'utiliser la validation croisée à k plis. Au départ, l'ensemble des données est divisé en deux parties : Un ensemble de tests et un autre ensemble qui est utilisé pour la validation croisée à k plis. En règle générale, 70 à 90 % des données sont destinées à l'apprentissage et le reste aux tests. Je choisis d'allouer 90 % à l'apprentissage car l'ensemble de données est relativement petit, pour s'assurer qu'il y a suffisamment de données dont on peut tirer des enseignements. Ces 90 % des données sont ensuite utilisés pour la validation croisée à k plis tandis que les 10 % de l'ensemble de tests sont mis de côté.

Ensemble de test	Ensemble de validation croisée à k plis	Nombre de plis (k)
10%	90%	5

Tableau 7 - Résumé de la division des données

Il faut être prudent lorsque l'on divise l'ensemble de données car on a des images différentes qui appartiennent au même patient et ces images sont fortement corrélées. Cela signifie que l'on doit éviter d'avoir une image d'un patient dans l'ensemble d'apprentissage et une autre image du même patient dans l'ensemble de validation ou de test, car ces images ne constitueraient pas des données vraiment nouvelles.

Pour y parvenir, j'ai développé un algorithme de partitionnement des données. Cet algorithme est basé sur un objet patient qui contient comme champ la liste des images correspondantes du patient. Certains patients n'ont qu'une seule image et d'autres en ont plusieurs. Les partitions ont cependant des tailles qui sont définies en termes d'images, par exemple, l'ensemble k-plis contient 90 % de toutes les images et l'ensemble test contient 10 %

de toutes les images. Comme expliqué précédemment, chaque patient ne doit appartenir qu'à une seule partition, donc chaque objet patient est attribué à une partition. Pour s'assurer que la taille correcte de chaque partition est atteinte, les objets patients sont mis en file d'attente en fonction du nombre d'images et les patients ayant le plus d'images sont mis en file d'attente en premier. Ensuite, le patient est attribué de manière aléatoire à une partition. La probabilité d'être attribué à une partition est proportionnelle au nombre d'espaces libres dans cette partition. Par conséquent, aucun patient ne sera attribué à une partition complète. En outre, comme les patients ayant une image sont les derniers dans la file d'attente, ils sont utilisés pour remplir précisément les partitions. Cette méthode de partitionnement des données est utilisée pour diviser l'ensemble de données d'origine en un ensemble de validation croisée et un ensemble de test, puis pour diviser l'ensemble de validation croisée en différents plis. Une des caractéristiques de cette approche de division des données est que la distribution des images COVID et des images non COVID n'est pas exactement égale. Ce fait doit être pris en compte plus tard lors de l'analyse des mesures de performance.

## **2.5) Algorithmes de recherche d'hyperparamètres**

Afin d'automatiser la recherche d'hyperparamètres, j'ai mis en œuvre deux algorithmes de recherche d'hyperparamètres : la recherche d'hyperparamètres par ensemble et la recherche d'hyperparamètres par grilles. Grâce à ces algorithmes, je rends le processus de sélection des hyperparamètres plus efficace en évitant de devoir modifier manuellement les hyperparamètres, de relancer l'exécution et de garder une trace de la performance de chaque ensemble d'hyperparamètres.

La recherche d'hyperparamètres par ensemble reçoit en entrée un ensemble d'objets hyperparamètres. L'objet hyperparamètres contient essentiellement comme champs une valeur pour chaque hyperparamètre. Les différents hyperparamètres sont :

- Rapport de division entre l'ensemble de tests et l'ensemble de validation croisée K-plis
- Nombre de plis
- Taille des lots
- Transformation d'apprentissage (ensemble de transformations qui sont appliquées l'ensemble d'apprentissage avant l'apprentissage)
- Transformation de validation (ensemble des transformations qui sont appliquées à l'ensemble de validation)
- Modèle
- Optimiseur
- Taux d'apprentissage
- Nombre d'époques
- Coefficient de régularisation L2
- Momentum
- Mode (RGB ou niveaux de gris)

- Nombre d'images utilisées pour l'évaluation multicrop

Tout le processus d'apprentissage est exécuté pour chaque objet hyperparamètres et certaines informations pertinentes sont enregistrées dans des fichiers. Par exemple, quelques mesures ainsi que le choix exact des hyperparamètres sont enregistrés dans un fichier texte. Chaque entrée de ce fichier texte correspond à une configuration particulière d'hyperparamètres, en d'autres termes, à un objet hyperparamètres et ils sont triés sur la base du MCC (coefficient de corrélation de Matthews). Cela signifie que la première entrée est, sur la base de ce critère, la combinaison d'hyperparamètres testés qui a donné les meilleurs résultats.

```
[mcc 0.59, acc 0.80, time 0h 13m 30.44s] : Model 24, trainVal_test_split = 0.9
transform = trainTransform RGB, val transform = trainTransform RGB, model = re
unfrozen layers = 4, width of classifier layers = [2048, 50], optimizer = RMSp
20, weight decay = 0.01, momentum = 0, color = RGB, n_crops = 1

[mcc 0.58, acc 0.79, time 0h 12m 52.64s] : Model 21, trainVal_test_split = 0.9
transform = trainTransform RGB, val transform = trainTransform RGB, model = re
unfrozen layers = 4, width of classifier layers = [2048, 200], optimizer = RMS
20, weight decay = 0.001, momentum = 0, color = RGB, n_crops = 1

[mcc 0.57, acc 0.79, time 0h 12m 59.61s] : Model 28, trainVal_test_split = 0.9
transform = trainTransform RGB, val transform = trainTransform RGB, model = re
unfrozen layers = 4, width of classifier layers = [2048, 100, 20], optimizer =
epochs = 20, weight decay = 0.01, momentum = 0, color = RGB, n_crops = 1

[mcc 0.57, acc 0.79, time 0h 13m 24.92s] : Model 25, trainVal_test_split = 0.9
transform = trainTransform RGB, val transform = trainTransform RGB, model = re
unfrozen layers = 4, width of classifier layers = [2048, 50], optimizer = RMSp
20, weight decay = 0.001, momentum = 0, color = RGB, n_crops = 1

[mcc 0.57, acc 0.79, time 0h 12m 55.02s] : Model 29, trainVal_test_split = 0.9
transform = trainTransform RGB, val transform = trainTransform RGB, model = re
unfrozen layers = 4, width of classifier layers = [2048, 100, 20], optimizer =
epochs = 20, weight decay = 0.001, momentum = 0, color = RGB, n_crops = 1
```

*Fig 13 - Fichier où sont enregistrées les configurations des modèles*

La matrice de confusion par époque et la perte pour chaque époque sont également enregistrées dans un autre fichier. À partir de ces informations, toutes les autres mesures et tracés peuvent être créés.

Cette fonction de recherche d'hyperparamètres par ensemble peut être utilisée pour créer une fonction de recherche d'hyperparamètres par grille, dans laquelle, au lieu de définir un ensemble d'objets hyperparamètres, on définit un ensemble de valeurs pour chaque hyperparamètre et l'algorithme parcourt toutes les combinaisons possibles.

## **2.6) Évaluation du modèle**

### **2.6.1) Évaluation multicrop**

Les données d'apprentissage sont recadrées de manière aléatoire avant d'être appliquées au réseau. Ceci est fait car les données sont composées d'images de tailles différentes mais qui doivent être de la même taille avant d'être appliquées au réseau. L'approche de recadrage permet d'éviter la distorsion qui serait causée par un simple redimensionnement à partir d'un rapport largeur/hauteur différent. Comme l'apprentissage est normalement exécuté sur plusieurs époques, différents recadrages d'une même image seront appliqués au réseau et on peut supposer que la plus grande partie de l'image est finalement vue par le réseau.

Pour la même raison, les images des ensembles de validation et de test sont également recadrées avant d'être appliquées au réseau. Cela introduit le problème que l'image entière n'est pas utilisée pour faire une prédiction. Pour avoir une évaluation plus complète du modèle, nous pouvons, pour chaque image, au lieu d'appliquer un seul recadrage au réseau, appliquer plusieurs recadrages aléatoires et prendre la moyenne des scores de tous les recadrages pour faire la prédiction finale. De cette façon, nous pouvons être sûrs qu'une plus grande partie de l'image est utilisée pour faire la prédiction. Cette évaluation multicrop n'est utilisée que pour la validation et les tests car, en apprentissage, cela équivaut à avoir simplement plus d'époques. Il existe une autre approche qui permet de s'assurer que l'image entière est utilisée pour l'évaluation, c'est ce qu'on appelle l'évaluation dense. Dans cette approche, toutes les couches doivent être convolutives, de sorte que les couches de classification sont converties en couches de convolution, ce qui est toujours possible. Un réseau entièrement convolutif peut recevoir des entrées de taille arbitraire, il suffit d'utiliser une couche de pooling moyenne à la fin pour obtenir une sortie de taille adéquate. Cette approche d'évaluation dense n'a pas été mise en œuvre. Ces deux techniques d'évaluation ont été tirées de l'article du VGG16 [10].

### **2.6.2) Visualisation des performances**

J'ai configuré le code de telle sorte que pendant l'exécution de l'apprentissage, certaines informations concernant le stade actuel de l'apprentissage et la consommation de temps sont affichées à l'écran.

fold: 0

```
Epoch: 0, Val time: 9.00s Train time: 55.97s,  
Epoch: 1, Val time: 4.18s Train time: 17.34s,  
Epoch: 2, Val time: 4.16s Train time: 17.20s,  
Epoch: 3, Val time: 4.16s Train time: 17.56s,  
Epoch: 4, Val time: 4.23s Train time: 17.59s,  
Epoch: 5, Val time: 4.20s Train time: 17.59s,  
Epoch: 6, Val time: 4.21s Train time: 17.60s,  
Epoch: 7, Val time: 4.19s Train time: 17.59s,  
Epoch: 8, Val time: 4.25s Train time: 17.87s,  
Epoch: 9, Val time: 4.22s Train time: 17.71s,
```

Total time: 0h 4m 20.84s

Fig 14 - Informations affichées pendant l'apprentissage

À la fin de chaque époque, la matrice de confusion et la perte pour l'ensemble d'apprentissage et l'ensemble de validation sont sauvegardées.

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Fig 15 - Illustration d'une matrice de confusion

À partir de la matrice de confusion, plusieurs autres mesures peuvent être calculées. Ces mesures et la perte peuvent être tracées en fonction de l'époque. Normalement, les mesures d'apprentissage et de validation seront tracées sur le même graphique pour qu'on puisse les comparer. Il existe également un graphique de la différence entre la perte de validation et la perte d'apprentissage pour étudier le surapprentissage. J'ai également décidé d'utiliser la surface sous cette différence de perte comme mesure numérique du surapprentissage. Cette valeur est affichée juste après la courbe précédente. Il y a régulièrement une certaine oscillation de la valeur d'une mesure d'une époque à l'autre, ce qui donne à ces graphiques un aspect bruyant. Ces oscillations à court terme entre les époques ne sont pas très pertinentes, c'est pourquoi tous les graphiques sont réalisés en appliquant une fonction de moyenne mobile aux valeurs afin de lisser la courbe et de pouvoir observer plus clairement le comportement à long terme.



Comme mentionné précédemment dans la section sur la division des données, un léger déséquilibre de classe pourrait se produire. Cela signifie que, par exemple, l'ensemble d'apprentissage peut contenir un peu plus d'images de patients COVID que de patients non COVID. En cas de déséquilibre entre les classes, la précision peut être une mesure trompeuse de la qualité du réseau. Par exemple, si 80 % des données de l'ensemble d'apprentissage appartiennent à une classe 0, un réseau qui classe chaque entrée dans la classe 0 atteindrait une précision de 80 %. La précision est relativement élevée, mais le réseau n'est pas un bon réseau et n'a rien appris des données. Le déséquilibre de classe présent dans nos ensembles de données n'est pas très important, donc cela ne devrait pas poser de problème grave. Quoi qu'il en soit, une mesure qui prend en considération ce déséquilibre des classes et produit une valeur plus fiable est le coefficient de corrélation de Matthew (MCC). Il s'agit donc de la principale mesure qui est utilisée pour classer les performances des différents réseaux.

Enfin, la Grad-CAM [11] est une technique utilisée pour produire des cartes thermiques qui sont superposées aux images d'entrée. Ces cartes thermiques montrent quelles zones des images ont été les plus significatives pour faire la prédiction et aident à comprendre plus intuitivement ce que le réseau regarde réellement afin d'effectuer certaines classifications. Cela a été utilisé pour aider à visualiser les décisions des réseaux.

## **2.7) Réduction du surapprentissage**

Comme l'ensemble de données est relativement petit, le surapprentissage est un problème important. Pour faire face à ce problème, il existe plusieurs stratégies.

### **2.7.1) Augmentation des données**

L'une des techniques les plus courantes pour traiter le surapprentissage lorsque les entrées sont des images est l'utilisation de l'augmentation des données. L'augmentation des données est simplement l'application de transformations aléatoires à l'entrée afin de créer de nouvelles données. A chaque époque, les mêmes images sont entrées dans le réseau, mais en raison de la nature aléatoire des transformations, chaque fois l'image est légèrement différente. Il y a beaucoup de transformations différentes qui peuvent être appliquées, mais nous devons être prudents et nous assurer que le résultat de la transformation ressemble toujours à une entrée crédible. Les transformations suivantes ont été utilisées pour l'augmentation des données :

- Retournement horizontal aléatoire
- Recadrage aléatoire
- Translation aléatoire

Ces transformations ont été choisies parce qu'elles produisent des résultats qui ressemblent encore à des tranches de CT scan thoracique transversale crédibles.

### **2.7.2) Régularisation**

La régularisation est une autre technique très traditionnelle pour faire face au surapprentissage. La régularisation tient compte du fait que le surapprentissage est souvent associé à des paramètres de grande ampleur. Par conséquent, la régularisation vise à diminuer la norme des paramètres en ajoutant un terme de pénalité à la fonction de perte qui doit être minimisée par la descente de gradient. Ce terme de pénalité est une somme de la norme des paramètres. La régularisation L2 est le type de régularisation le plus courant et dans ce type de régularisation, la norme L2 est utilisée. Le terme de pénalité a un coefficient qui est utilisé pour ajuster dans quelle mesure la norme des paramètres doit être minimisée.

### **2.7.3) Couches dropout**

L'un des principaux problèmes de l'apprentissage des grands réseaux est la co-adaptation. Dans un tel réseau, si tous les poids sont appris ensemble, il est courant que certaines connexions aient une capacité prédictive plus importante que d'autres. Au fur et à mesure que le réseau est entraîné de manière itérative, ces connexions puissantes sont davantage apprises tandis que les plus faibles sont ignorées. Au cours de nombreuses itérations, seule une fraction des connexions de nœuds est entraînée et les autres cessent de participer. Les couches dropout sont des couches qui sont introduites entre des couches du perceptron multicouche et qui annulent aléatoirement la sortie des nœuds. Cette procédure résout le problème de la co-adaptation.

### **2.7.4) L'utilisation de modèles pré-entraînés**

Une autre stratégie très puissante pour faire face au surapprentissage, en particulier dans le cas d'un petit ensemble de données d'image, est l'utilisation d'un réseau convolutionnel pré-entraîné. Un réseau pré-entraîné est un réseau qui a déjà été formé sur un ensemble de données pour une tâche particulière. L'utilisation d'un réseau pré-entraîné signifie la réutilisation de ce réseau avec les paramètres appris, pour notre problème spécifique. Ces paramètres appris sont bloqués et ne changent pas pendant l'apprentissage. Toutefois, les paramètres du classifieur et les paramètres des dernières couches de convolution sont débloqués et sont mis à jour pendant l'apprentissage. Une des caractéristiques de la classification d'images est que les motifs de bas niveau à partir desquels les motifs plus complexes et abstraits sont créés sont les mêmes pour la plupart des problèmes. Ce sont les motifs complexes qui varient d'un problème à l'autre. Cela

signifie que si l'on dispose d'un ensemble de données suffisamment important à partir duquel on peut apprendre ces motifs de bas niveau utiles, alors on peut former un réseau sur cet ensemble de données et ensuite réutiliser, au moins les premières couches du réseau, sur tout problème similaire. Il existe un très grand ensemble de données d'objets et d'animaux de la vie quotidienne connu sous le nom d'ImageNet et les architectures de réseaux neuronaux les plus populaires sont disponibles pré-entraînés sur cet ensemble de données. Si l'on formait notre réseau uniquement sur les données dont on dispose, il n'y aurait peut-être pas assez de données pour l'apprentissage de ces motifs de bas niveau. Néanmoins, ces motifs pourraient toujours être utiles pour la classification correcte des entrées. C'est pourquoi une bonne stratégie consiste à prendre ces réseaux pré-entraînés sur imageNet, à ajouter un nouveau classifieur et à débloquer les dernières couches de la base convolutive. On peut remarquer que le fait que certains paramètres soient bloqués et non mis à jour pendant l'apprentissage réduit le nombre réel de paramètres à apprendre. En réduisant le nombre de paramètres, nous pouvons réduire le surapprentissage.

## **2.8) Modèles**

Plusieurs réseaux neuronaux convolutifs populaires ont été étudiés. Dans chaque cas, les algorithmes de recherche d'hyperparamètres mentionnés précédemment ont été utilisés pour aider à trouver les hyperparamètres qui produisent les meilleurs résultats. Ces réseaux ont tous été pré-entraînés sur l'ensemble ImageNet comme mentionné auparavant. Seule la base convolutive des modèles a été réutilisée, car le classifieur doit être modifié pour s'adapter à notre problème.

Les réseaux pré-entraînés attendent des images RGB en entrée. Afin de pouvoir fournir des images en niveaux de gris, la première couche doit être ajustée. Cela se fait en remplaçant les trois canaux des noyaux de la première couche par des noyaux équivalents à un canal, dans lesquels les éléments sont la somme des éléments des trois canaux.

Comme tous les réseaux sont pré-entraînés sur ImageNet, ils attendent des images qui présentent certaines caractéristiques. C'est pourquoi la normalisation selon ImageNet est appliquée aux images avant l'apprentissage. Les canaux sont normalisés de manière à avoir les moyennes suivantes : 0,485, 0,456, 0,406 et les variances entre les canaux sont normalisés de manière à obtenir les valeurs suivantes : 0,229, 0,224, 0,225. Cette normalisation est recommandée par torchvision, qui est la bibliothèque à partir de laquelle ces modèles pré-entraînés sont obtenus.

### 2.8.1) VGG16

La première architecture de réseau qui a été étudiée est le réseau VGG16 [10], qui est un réseau neuronal convolutif très traditionnel. La base convolutionnelle du réseau VGG16 que l'on va utiliser présente les caractéristiques suivantes. L'entrée est une image RGB de taille fixe de  $224 \times 224$  qui est passée à travers une série de couches convolutionnelles, où des filtres avec des noyaux de taille  $3 \times 3$  sont utilisés avec un pas de 1. Le padding spatial de l'entrée des couches convolutionnelles est tel que la résolution spatiale est préservée après la convolution. Le pooling spatial est effectué par cinq couches max pooling, qui suivent certaines des couches de convolution. Le max pooling est effectuée sur une fenêtre de  $2 \times 2$  pixels, avec un pas de 2. Toutes les couches cachées sont équipées des fonctions d'activation ReLU.

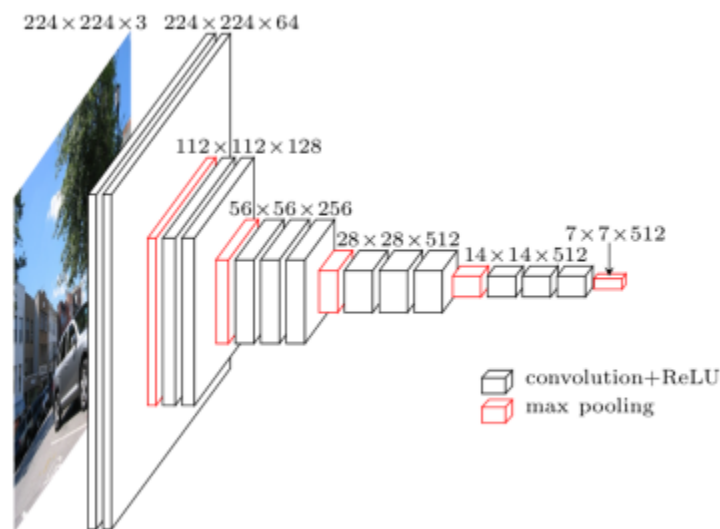


Fig 16 - Architecture VGG16

Grâce à l'expérimentation et à l'aide des algorithmes de recherche d'hyperparamètres développés, les hyperparamètres qui offrent les meilleures performances pour ce réseau sont les suivants :

Taille de lots	Taux d'apprentissage	Nombre d'époques	Optimiseur
16	0,00001	5	RMSprop
Coefficient de Régularisation L2	Nombre d'images (évaluation multicrop)	Largeurs des couches du classifieur	Nombre de couches non bloquées
0,01	3	128	4

Tableau 8 - Hyperparamètres choisis pour l'architecture VGG16

Ce réseau avec ces hyperparamètres produisent les résultats suivants :

<b>TN</b>	<b>FP</b>	<b>FN</b>	<b>TP</b>	<b>Perte</b>
82%	18%	23%	77%	0,45
<b>Exactitude</b>	<b>MCC</b>	<b>Précision</b>	<b>Rappel</b>	<b>Différence de perte</b>
80%	0,59	77%	77%	0,09

*Tableau 9 - Performances de l'architecture VGG16*

Des graphiques détaillés, qui contiennent également des informations sur les mesures relatives à l'ensemble d'apprentissage, sont disponibles dans l'annexe 2.

## 2.8.2) ResNet-50

Traditionnellement, plus un réseau est profond, mieux c'est, mais avec les simples ajouts de couches, nous observons souvent une dégradation des performances due au problème de la disparition du gradient, dans lequel le gradient qui se propage par rétropropagation lors de la descente stochastique du gradient devient trop petit. Les ResNets ou réseaux résiduels [12] résolvent ce problème par l'introduction de connexions de raccourci qui relient la sortie d'une couche à l'entrée d'une autre couche qui se trouve deux ou trois couches devant. Les couches sont normalement connectées l'une après l'autre et ces connexions de raccourci sont supplémentaires. Par conséquent, la sortie d'une connexion de raccourci est ajoutée à l'entrée régulière d'une couche, ce qui signifie qu'elles doivent avoir la même taille. Si ce n'est pas le cas, alors la sortie de la connexion de raccourci doit être projetée à la bonne taille par des convolutions de 1x1 ou elle peut être complétée par des zéros pour atteindre la bonne taille. Ces connexions de raccourcis n'introduisent normalement pas de nouveaux paramètres puisqu'il s'agit de mappages d'identité. Cependant, si les tailles ne correspondent pas et que des convolutions 1x1 sont utilisées pour la projection, alors les coefficients des noyaux de ces convolutions introduisent de nouveaux paramètres.

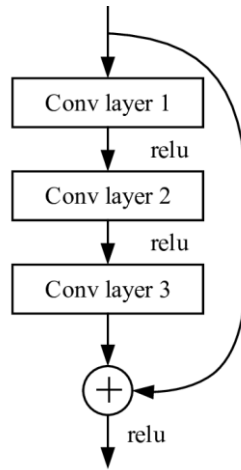


Fig 11 - Connexion raccourcie dans les ResNets

Il existe plusieurs versions de ResNets, j'ai étudié l'utilisation du ResNet-50, qui contient 50 couches, afin de voir s'il y a des avantages à utiliser un réseau plus profond.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$

Fig 12 - Architectures ResNet

### 2.8.2.1) Normalisation par lots

Les couches de normalisation par lots [13] sont des couches qui précèdent les couches de convolution ou des couches des perceptrons multicouche et qui normalisent les données d'un lot. Cette normalisation consiste à centrer la moyenne sur 0 et à mettre la variance à 1. La normalisation par lots rend les réseaux plus rapides et plus stables, mais la raison exacte est encore discutée. Ce type de couche n'est pas spécifique aux réseaux résiduels, mais est couramment utilisé dans ces réseaux.



Les hyperparamètres qui fournissent les meilleures performances pour ce réseau sont les suivants :

Taille de lots	Taux d'apprentissage	Nombre d'époques	Optimiseur
16	0,00001	12	RMSprop
Coefficient de Régularisation L2	Nombre d'images (évaluation multicrop)	Largeurs des couches du classifieur	Nombre de couches non bloquées
0,01	3	256	4

Tableau 10 - Hyperparamètres choisis pour l'architecture ResNet-50

Ce réseau avec ces hyperparamètres donne les résultats suivants :

TN	FP	FN	TP	Perte
83%	17%	25%	75%	0,50
Exactitude	MCC	Précision	Rappel	Différence de perte
79%	0,58	79%	75%	0,06

Tableau 11 - Performances de l'architecture ResNet-50

Des graphiques détaillés, qui contiennent également des informations sur les mesures relatives à l'ensemble d'apprentissage, sont disponibles dans l'annexe 2.

### 2.8.3) DenseNet-169

Les DenseNets [14], tout comme les ResNets, utilisent également des connexions de raccourcis. Ils sont constitués de deux blocs, les blocs denses et les blocs de transition. Les blocs denses contiennent des couches de convolution et des couches de normalisation par lots, et la sortie d'une couche de convolution est l'entrée de chaque couche de convolution suivante du bloc. Ces entrées des couches précédentes sont toutes concaténées avant d'être appliquées à une couche, contrairement aux ResNets, où elles sont additionnées. Les couches de transition contiennent les couches de convolution et les couches pooling. L'un des avantages de cette architecture est que les motifs de bas niveau se déplacent plus profondément dans le réseau et sont utilisés par les couches plus profondes, en même temps que les motifs plus complexes. Ces connexions de raccourcis résolvent également le problème de la disparition des gradients, comme les ResNets, et permettent donc l'utilisation de réseaux plus profonds.

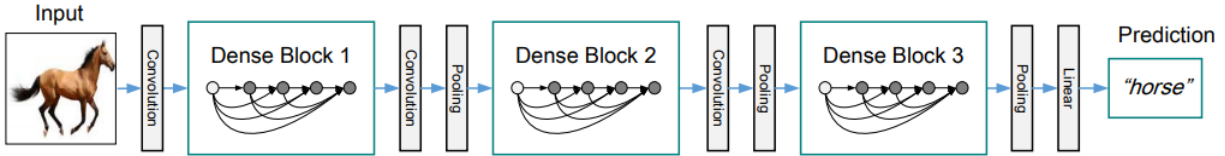


Fig 17 - DenseNet

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	$112 \times 112$	$7 \times 7$ conv, stride 2			
Pooling	$56 \times 56$	$3 \times 3$ max pool, stride 2			
Dense Block (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$56 \times 56$	$1 \times 1$ conv			
	$28 \times 28$	$2 \times 2$ average pool, stride 2			
Dense Block (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$28 \times 28$	$1 \times 1$ conv			
	$14 \times 14$	$2 \times 2$ average pool, stride 2			
Dense Block (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	$14 \times 14$	$1 \times 1$ conv			
	$7 \times 7$	$2 \times 2$ average pool, stride 2			
Dense Block (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$

Fig 18 - Architectures DenseNet

Les hyperparamètres qui offrent les meilleures performances pour ce réseau sont les suivants :

Taille de lots	Taux d'apprentissage	Nombre d'époques	Optimiseur
16	0,0001	10	RMSprop
Coefficient de Régularisation L2	Nombre d'images (évaluation multicrop)	Largeurs des couches du classifieur	Nombre de couches non bloquées
0,001	3	128	4

Tableau 12 - Hyperparamètres choisis pour l'architecture Densenet-169

Ce réseau avec ces hyperparamètres produisent les résultats suivants :

TN	FP	FN	TP	Perte
87%	13%	18%	82%	0,36
Exactitude	MCC	Précision	Rappel	Différence de perte
84%	0,68	83%	82%	0,16

Tableau 13 - Performances de l'architecture Densenet-169

Des graphiques détaillés, qui contiennent également des informations sur les mesures relatives à l'ensemble d'apprentissage, sont disponibles dans l'annexe 2.

## 2.8.4) Comparaison Grad-CAM

La figure suivante montre le résultat de la visualisation grad-CAM pour les trois modèles testés pour quelques images de l'ensemble de test.

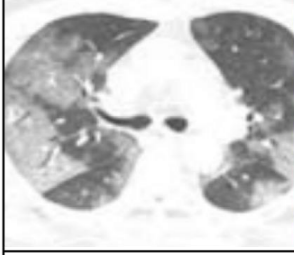
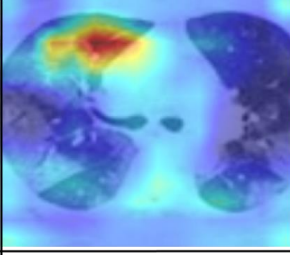
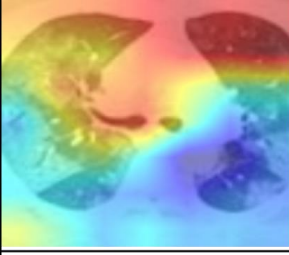
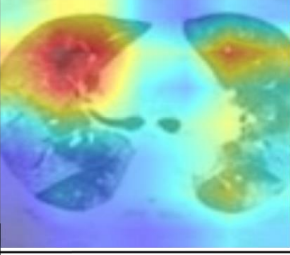
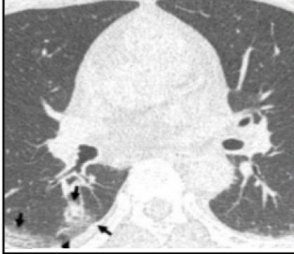
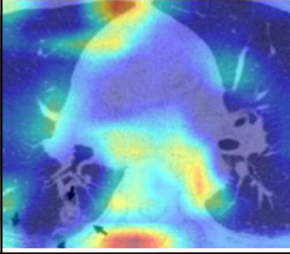
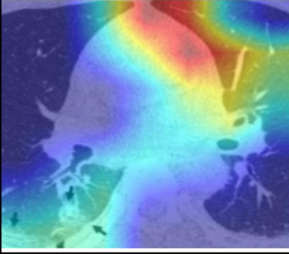
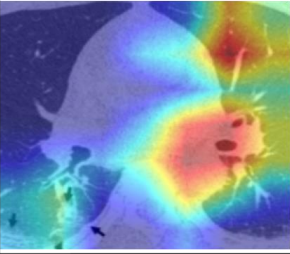
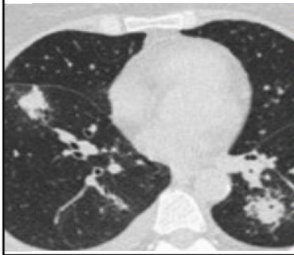
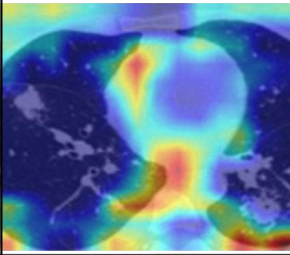
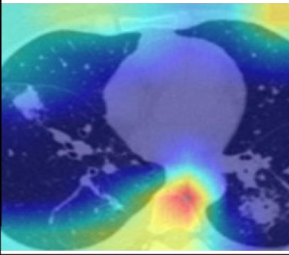
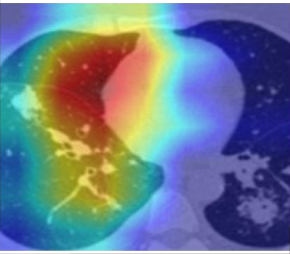
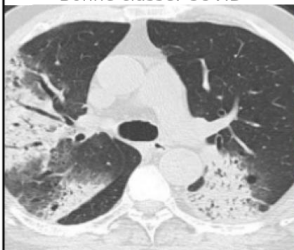
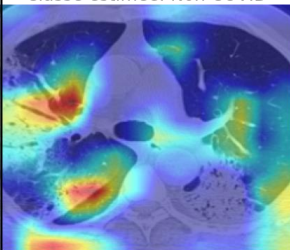
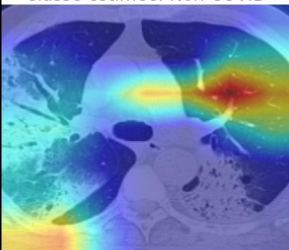
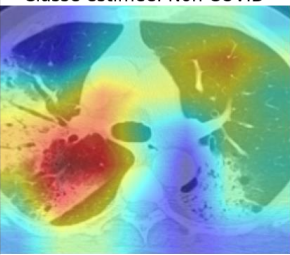
	VGG-16	ResNet-50	DenseNet-169
Bonne classe: Non COVID	Classe estimée: Non COVID	Classe estimée: Non COVID	Classe estimée: Non COVID
			
Bonne classe: COVID	Classe estimée: Non COVID	Classe estimée: COVID	Classe estimée: COVID
			
Bonne classe: COVID	Classe estimée: Non COVID	Classe estimée: Non COVID	Classe estimée: COVID
			
Bonne classe: COVID	Classe estimée: Non COVID	Classe estimée: Non COVID	Classe estimée: Non COVID
			

Fig 19 - Grad-cam pour les trois modèles testés

### 3) Analyse des résultats

En examinant les valeurs des paramètres utilisés, on peut remarquer que le taux d'apprentissage varie entre 0,00001-0,0001. L'optimiseur RMSprop, qui a été utilisé dans tous les cas, modifie le taux d'apprentissage de manière dynamique, de sorte que ces valeurs précédentes ne représentent que le taux d'apprentissage initial. Le nombre d'époques reste relativement faible, entre 5 et 12. Ces valeurs ont été déterminées en fonction du moment où un surapprentissage important a commencé à apparaître. Le coefficient de régularisation L2 varie entre 0,001-0,01. Bien que, la plupart du temps, il n'y aurait pas de changements significatifs si le coefficient de régularisation L2 passait d'une de ces valeurs à l'autre. Un phénomène que j'ai observé est que les coefficients de régularisation L2 les plus faibles étaient associés à des courbes plus lisses. Le nombre d'images utilisées pour l'évaluation multicrop était 3 dans tous les cas. L'utilisation d'images multiples est utile car elle introduit de nouvelles informations, mais finalement ces nouvelles images n'introduisent pas d'informations significatives mais contribuent quand même à la complexité temporelle de l'algorithme. Enfin, la largeur de la couche intermédiaire du classifieur (avant la couche de sortie) variait entre 128 et 256 et le nombre de couches non bloquées à la fin de la base convolutive était de 4. Ces deux paramètres affectent la capacité du réseau (le nombre de paramètres à apprendre), ils doivent donc être ajustés ensemble, ainsi que le coefficient de régularisation L2 afin de produire de bons résultats de validation et en même temps des niveaux de surapprentissage peu significatifs. J'ai expérimenté l'utilisation d'un plus grand nombre de couches dans le classifieur, ce qui a semblé réduire considérablement le surapprentissage, mais en même temps a diminué les performances d'apprentissage et de validation. La largeur du classifieur est un multiple de 2 car il s'agit d'une heuristique classique.

Compte tenu des performances de validation, si l'on devait classer les modèles sur la base d'une mesure, on utiliserait la MCC. Dans ce cas, DenseNet-169 (0,68) produit les meilleurs résultats, suivi de VGG16 (0,59) et ensuite de ResNet-50 (0,58). Le même classement serait obtenu si l'on utilisait l'exactitude (84%, 80%, 79%), ce qui montre que le déséquilibre de classe dans les données n'est pas très significatif. De plus, les résultats produits par VGG16 et ResNet-50 sont très similaires, je dirais donc qu'ils sont à égalité pour la deuxième place au lieu de dire que l'un a clairement surpassé l'autre. Pour évaluer le surapprentissage, on peut regarder la différence de perte, qui mesure la différence entre la perte de validation et la perte d'apprentissage. Dans ce cas, ResNet-50 (0,06) est le réseau le moins surappris, suivi de VGG16 (0,09) et ensuite de DenseNet-169 (0,16). Toutefois, la perte globale est plus faible pour le DenseNet-169. En ce qui concerne le taux de vrais négatifs et le taux de faux positifs, c'est-à-dire le diagnostic correct d'une personne qui n'a pas de COVID-19, DenseNet-169 (87%/13%) donne les meilleurs résultats, suivi de ResNet-50 (83%/17%), qui est légèrement meilleur que VGG16 (82%/18%). En ce qui concerne le taux de vrais positifs et le taux de faux négatifs, c'est-à-dire le diagnostic correct d'une personne atteinte de la COVID-19, DenseNet-169 (82 %/18 %) donne les meilleurs résultats, suivi de VGG16 (77 %/23 %) et ensuite de ResNet-50 (75 %/25 %). Dans l'ensemble, nous pouvons observer que chaque réseau obtient de meilleures performances dans le diagnostic d'une personne non atteinte de la COVID-19 par rapport à une

personne atteinte de la COVID-19. 87%/82% pour DenseNet-169, 83%/75% pour ResNet -50 et 82%/77% pour VGG16.

En analysant les résultats de la visualisation Grad-CAM, on remarque que parfois les annotations sont prises en compte par l'algorithme. Pour résoudre ce problème, il faudrait utiliser une méthode permettant de supprimer les annotations ou un ensemble de données sans distorsion. On remarque également que parfois les tissus qui ne constituent pas les poumons (grandes régions blanches) sont parfois utilisés par le classifieur. Une approche pour résoudre ce problème consisterait à appliquer certaines techniques de segmentation pour isoler uniquement les poumons.

## **Conclusion**

Ce travail visait à résoudre deux problèmes : Le premier était de produire une méthode alternative de diagnostic de la COVID-19 en apprenant à un ordinateur comment analyser les CT scans des patients. Le second problème était de produire un ensemble de données suffisamment important et diversifié, utile pour cet apprentissage. C'est un défi car ces données médicales sont souvent inaccessibles au public. Zhao et al [8] proposent une approche créative pour créer cet ensemble de données en extrayant des images d'articles scientifiques, ce qui a été l'inspiration pour ce travail. Ces images, cependant, contiennent beaucoup de distorsions et ont été soigneusement analysées au début de ce travail afin d'identifier les différentes sources de distorsion. Certaines de ces distorsions ont été traitées et d'autres non. Par exemple, la présence d'annotations dans les images n'a pas été traitée dans ce travail, mais pourrait être traitée à l'avenir afin de produire des images plus propres. Une approche possible serait d'utiliser l'inpainting pour supprimer ces annotations et les remplacer par quelque chose qui est susceptible d'exister réellement dans l'image. Cette procédure nécessiterait toutefois une identification manuelle des pixels de l'annotation, à moins que d'autres techniques ne soient proposées pour identifier les pixels des annotations. Outre le fait qu'il contenait des images déformées, l'ensemble de données était relativement petit et contenait plusieurs images du même patient, ce qui posait des problèmes lors de l'évaluation des performances, car l'ensemble de validation pouvait ne pas être représentatif de la population. Le premier problème a été résolu par la conception d'un algorithme pour la division de l'ensemble de données et le second par l'utilisation d'une validation croisée à k plis. La validation croisée k-plis produit des mesures plus fiables mais a également augmenté de manière significative le temps d'exécution, ce qui a limité le nombre de tests pouvant être exécutés.

En ce qui concerne le premier problème, la solution a été d'appliquer le deep learning afin d'apprendre à partir de l'ensemble des données produits. L'une des principales difficultés du deep learning est qu'il existe tellement de possibilités différentes qui pourraient être essayées qu'il est difficile de découvrir quelle configuration devrait produire les meilleurs résultats. C'est pourquoi j'ai décidé d'expérimenter trois modèles très populaires qui produisent de bons résultats de manière



constante dans différentes tâches : VGG16, ResNet-50 et DenseNet-169. Il y a encore beaucoup d'hyperparamètres à choisir et pour faciliter ce processus, j'ai développé des algorithmes de recherche automatique d'hyperparamètres. Cependant, ces algorithmes ne peuvent pas faire tout le travail, car il n'existe pas de mesure à valeur unique pouvant être utilisée pour classer les performances des réseaux de neurones. J'ai donc également mis en œuvre plusieurs fonctions pour extraire des mesures des résultats et produire différents graphiques. Par exemple, il y a la fonction Grad-CAM qui a été utilisée pour produire une compréhension plus intuitive de ce que le réseau fait réellement, ce qui est utile pour identifier l'origine des erreurs et proposer des idées pour l'amélioration du réseau. Un autre problème très important dans le deep learning est le surapprentissage. Le facteur le plus important de ce problème est probablement la petite taille de l'ensemble de données. Pour faire face à cet obstacle, certaines stratégies ont été adoptées, telles que la régularisation L2, les couches dropout, l'augmentation des données et l'utilisation de modèles pré-entraînés.

Les performances des modèles sont prometteuses, la meilleure configuration atteignant une précision d'environ 84%. Cela n'est évidemment pas suffisant pour être appliqué dans la pratique, mais indique que cette approche, avec plus de développement, pourrait à terme concurrencer les tests utilisés aujourd'hui. Quelques idées pour améliorer ces performances sont : l'utilisation d'un ensemble de données plus important avec des images non déformées, l'utilisation d'une technique de segmentation pour isoler les poumons et l'emploi d'architectures de réseaux neuronaux de pointe.



## **Annexes**

### **Annexe 1 - Théorie des réseaux de neurones [15-19]**

#### **Les réseaux de neurones artificiels**

Les réseaux neuronaux artificiels, normalement appelés simplement réseaux de neurones (RN), sont des systèmes informatiques vaguement inspirés des cerveaux d'animaux. Un réseau de neurones est essentiellement un ensemble d'unités ou de nœuds connectés appelés neurones artificiels. Chaque connexion, comme les synapses d'un cerveau biologique, peut transmettre un signal à d'autres neurones. Un neurone artificiel qui reçoit un signal le traite et peut signaler les neurones qui lui sont connectés. Les neurones et les connexions ont généralement un poids qui s'ajuste au fur et à mesure de l'apprentissage. Les poids augmentent ou diminuent l'intensité du signal au niveau d'une connexion.

En général, les neurones sont regroupés en couches. Différentes couches peuvent effectuer des opérations différentes sur leurs entrées. Les signaux passent de la couche d'entrée à la couche de sortie, en traversant normalement des couches intermédiaires appelées couches cachées. Les types de couches et la façon dont elles sont connectées sont appelés architecture du réseau neuronal, le nombre de couches est connu comme la profondeur du réseau et le nombre de neurones dans une couche est connu comme la largeur de la couche. Le principal travail dans un problème de deep learning est de choisir une bonne architecture et de trouver la profondeur et la largeur appropriées des couches.

Les couches cachées peuvent être définies arbitrairement, mais ce n'est pas le cas pour la couche de sortie. La couche de sortie est toujours déterminée par le type de problème car elle doit produire une sortie appropriée. Dans un problème de classification, il y a autant de neurones dans la couche de sortie qu'il y a de classes. Chaque neurone produira un nombre qui est un score associé à la probabilité que l'entrée originale appartienne à une classe particulière. En outre, ces scores peuvent être passés par une fonction d'activation, comme un sigmoïde ou un softmax, afin de les convertir en probabilités. Pour décider de la prédiction finale du modèle, vous devez examiner tous les scores ou probabilités et sélectionner la valeur la plus élevée.

#### **Réseau à propagation directe (feedforward)**

Un réseau à propagation directe est un réseau neuronal dans lequel chaque couche se connecte uniquement aux couches suivantes, en d'autres termes, les connexions ne forment pas de cycles et l'information se déplace dans une direction, de l'entrée à la sortie.

#### **Perceptron multicouche**

Un perceptron multicouche est un type de réseau neuronal à propagation directe dans lequel chaque neurone artificiel est un perceptron. Un perceptron peut être considéré comme une unité de calcul qui possède plusieurs connexions d'entrée et chaque connexion a un poids associé. Chaque entrée du perceptron est multipliée par le poids correspondant et additionnée

avec un terme de biais. Le résultat passe ensuite par une fonction d'activation non linéaire et constitue la sortie du perceptron, qui est à son tour l'entrée des perceptrons de la couche suivante.

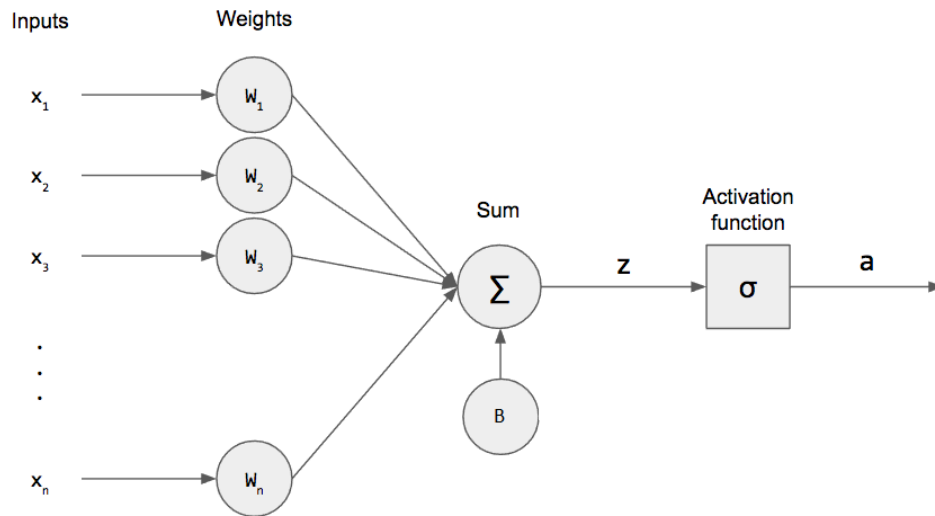


Fig 20 - Perceptron

Un perceptron multicouche est également entièrement connecté, ce qui signifie que chaque neurone d'une couche est connecté à tous les autres neurones de la couche suivante.

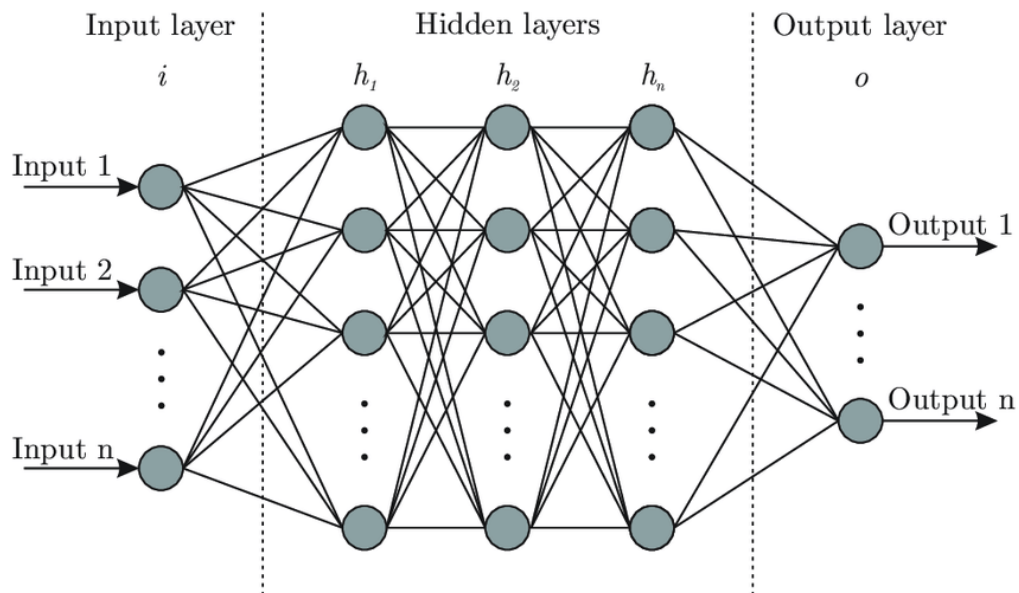


Fig 21 - Perceptron multicouche (chaque nœud est un perceptron)

Les poids des connexions et les biais des neurones sont appelés les paramètres à apprendre du réseau. Au début, ces paramètres sont définis de manière aléatoire, mais au fur et à mesure que l'apprentissage progresse, ils sont automatiquement ajustés pour augmenter les performances du réseau sur les données dont il tire des apprentissages. Le nombre total de paramètres pouvant être appris d'un réseau neuronal est désigné par le terme de capacité.

## Couche de convolution

Une couche de convolution est une couche composée de neurones convolutifs. Chaque neurone convolutif a un noyau associé et un terme de biais. Dans une opération de convolution, l'entrée est une image et la sortie est une autre image. Les pixels de la nouvelle image sont déterminés un à la fois en faisant glisser le noyau de convolution autour de l'image et en multipliant les éléments correspondants, en les ajoutant au terme de biais et en faisant passer le résultat par une fonction d'activation. La taille du noyau est définie par l'utilisateur, tout comme le pas, c'est-à-dire le nombre de pixels qui sont sautés lorsque le noyau glisse sur l'image. Le nombre de canaux du noyau est déterminé par le nombre de canaux de son entrée.

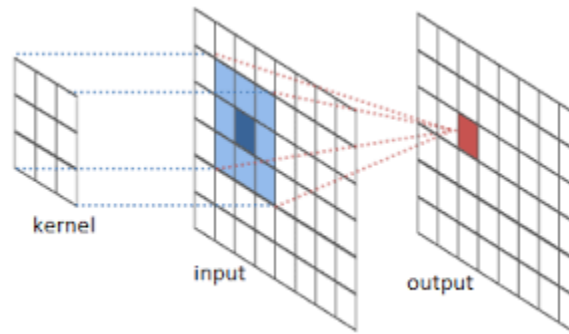


Fig 22 - Opération de convolution

Un perceptron a autant de poids qu'il y a de perceptrons dans la couche précédente plus un biais. Un neurone convolutif, en revanche, a autant de poids que la taille de son noyau plus un biais. Comme le nombre de perceptrons par couche peut être important et que la taille des noyaux convolutifs est souvent petite, un neurone convolutif a normalement beaucoup moins de paramètres à apprendre qu'un perceptron.

L'opération de convolution a plusieurs applications dans le traitement de l'image, comme l'ajout de flou, la détection des contours et l'accentuation de la netteté. Elle peut également être interprétée comme une détection de caractéristiques ou de motifs, dans laquelle le motif détecté est celui créé par les coefficients du noyau. C'est le cas puisque le résultat d'une opération de convolution aura des valeurs élevées ou des pixels brillants aux endroits où le motif du noyau est le plus important dans l'image originale. Dans le contexte des réseaux de neurones, le résultat d'une opération de convolution est connu sous le nom de carte de caractéristiques, car il informe de l'endroit où la caractéristique du noyau correspondant est la plus fréquente. Dans une couche de convolution, qui comporte plusieurs neurones, chaque neurone détectera des motifs différents.

La première couche convolutive détecte des motifs simples et passe les cartes de caractéristiques à la couche suivante. Comme la couche suivante reçoit les cartes de caractéristiques en entrée, elle détecte des motifs composés de motifs plus petits. De cette façon, plus nous nous approfondissons dans les couches, plus les motifs deviennent complexes. C'est pourquoi on dit que les réseaux neuronaux convolutifs apprennent les modèles de façon hiérarchique. Les poids et les biais d'une couche convolutive sont initialement aléatoires, tout comme pour les perceptrons multicouches, ce qui signifie qu'ils détectent des motifs aléatoires. Cependant, au fur et à mesure de l'apprentissage, ces poids et biais sont automatiquement ajustés de manière à ce que le motif détecté soit réellement utile.

## Couche de pooling

Les couches de pooling sont responsables de la réduction de la dimension spatiale des cartes de caractéristiques, en d'autres termes, la largeur et la hauteur. L'opération de pooling est exécutée par le déplacement d'une fenêtre sur l'image, de manière similaire à l'opération de convolution. Cependant, cette fenêtre n'a pas de coefficients, son but est de remplacer des parties de l'image par un pixel correspondant. Le max pooling est le type de pooling le plus utilisé, il substitue des zones de l'image par le pixel de valeur maximale à l'intérieur de la zone. Il existe également le pooling moyen, qui consiste à remplacer des zones de l'image par la moyenne des pixels de la zone. Le pooling est une opération utile parce que l'on perd la position exacte d'un motif et que l'on commence à ne connaître que la région globale où le motif a été identifié. Cela crée une invariance de translation et permet de combiner des motifs légèrement éloignés par les couches de convolution plus profondes. Les couches de pooling n'ont pas de paramètres à apprendre et exécutent toujours une opération fixe. En réduisant la taille de l'image, les couches de pooling augmentent également la vitesse des réseaux.

## Réseaux de neurones convolutifs

Les réseaux neuronaux convolutionnels complètent les perceptrons multicouches avec l'inclusion de blocs de convolution. Ces blocs de convolution sont constitués de couches de convolution et de pooling. Un réseau neuronal convolutif se compose souvent de deux parties : une base convolutive et un classifieur. La base convolutive est normalement une séquence de blocs de convolution et le classifieur est un perceptron multicouche. La base convolutionnelle est responsable de la détection de caractéristiques abstraites et le classifieur reçoit ces cartes de caractéristiques et est chargé d'analyser la présence et la position des caractéristiques afin d'effectuer une classification finale de l'image d'entrée originale.

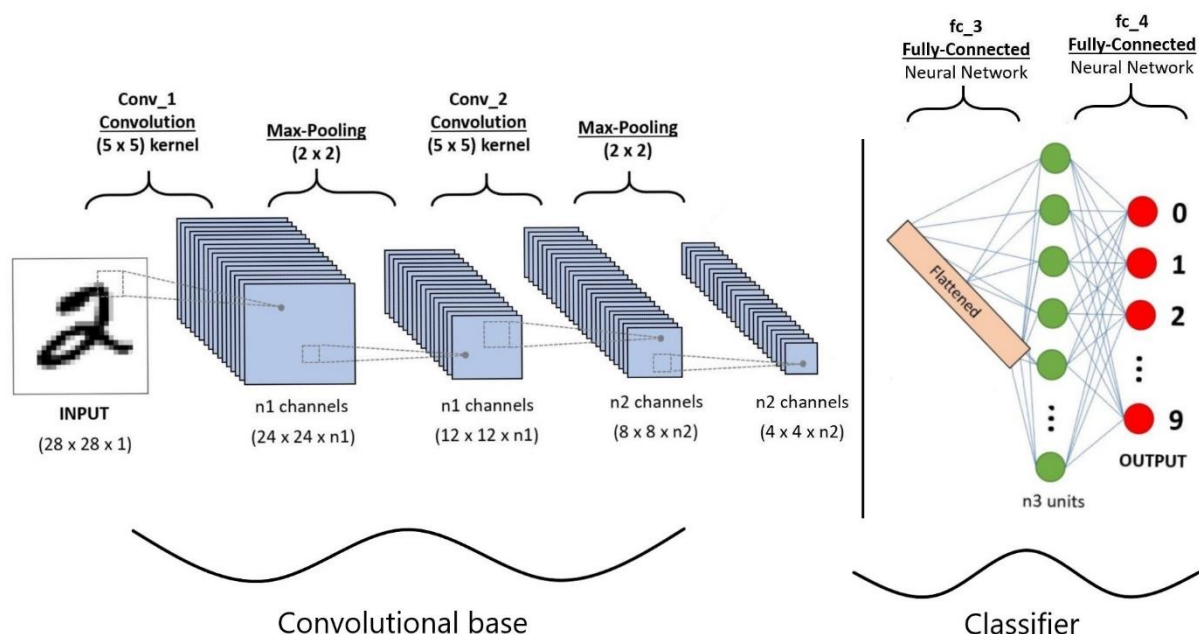


Fig 23 - Architecture standard de réseau neuronal convolutif

## Apprentissage

L'apprentissage d'un réseau de neurones est le processus par lequel il apprend des données et consiste à ajuster les poids et les biais de telle sorte que la sortie soit cohérente. Au départ, une entrée provenant de l'ensemble de données est introduite dans le réseau, ce qui produit une certaine sortie. Comme les poids et les biais sont initialisés de manière aléatoire, ce résultat sera très probablement incorrect. Cependant, comme nous sommes dans un scénario d'apprentissage supervisé, nous savons déjà ce que la sortie devrait être. Par conséquent, nous pouvons comparer les deux résultats et, en outre, nous pouvons utiliser une fonction de perte appropriée pour quantifier la distance qui les sépare. La fonction de perte est déterminée par le type de problème. Pour les problèmes de classification binaire, comme le nôtre, la fonction de perte standard est l'entropie croisée catégorielle. Notre objectif est évidemment de diminuer la perte, puisque cela signifie que la sortie produite par le réseau est proche de la sortie correcte. Il s'avère en fait que nous pouvons exprimer la perte en fonction des poids et des biais du réseau.

## Algorithme du gradient

Cela nous permet de voir notre problème comme un problème d'optimisation, dans lequel la fonction de perte est la fonction objective que nous voulons minimiser et les entrées de cette fonction sont les paramètres à apprendre. Ce problème peut être résolu numériquement grâce à l'algorithme du gradient. Dans l'algorithme du gradient, nous commençons à un point dans le domaine de la fonction objectif. Ce point sera aléatoire puisqu'il correspond à un ensemble de paramètres à apprendre qui sont arbitrairement définis au départ. Ensuite, nous calculons le gradient de la fonction objectif par rapport à ses variables au point courant. Nous savons que le négatif du gradient pointe dans la direction de la plus grande diminution et nous nous déplaçons donc du point actuel dans cette direction, dans le but de minimiser la fonction objectif. L'ampleur de notre mouvement est appelée le taux d'apprentissage et doit être définie avant le début de l'apprentissage. Cette procédure est répétée dans le but d'atteindre la valeur minimale de la fonction objectif. Nous atteindrons finalement un minimum global si la fonction objectif est convexe ou un minimum local. Dans ce dernier cas, il existe différentes stratégies pour échapper à un minimum local qui n'est pas le minimum global.

## Rétropropagation

L'une des difficultés de cette approche est le calcul du gradient de la fonction de perte par rapport aux paramètres à apprendre. Ce problème est résolu par l'algorithme de rétropropagation, qui a été une découverte importante dans le domaine du machine learning. L'algorithme de rétropropagation fonctionne en calculant le gradient de la fonction de perte par rapport à chaque poids selon la règle de la chaîne, en calculant le gradient une couche à la fois, en faisant des itérations à partir de la dernière couche pour éviter des calculs redondants de termes intermédiaires dans la règle de la chaîne.

## L'algorithme du gradient stochastique

L'utilisation de l'algorithme de rétropropagation permet de calculer le gradient et d'actualiser les paramètres à apprendre en fonction de l'algorithme du gradient afin de diminuer la perte. Dans l'explication précédente, par souci de simplicité, j'ai considéré qu'une entrée de l'ensemble de données était introduite dans le réseau à la fois. Cela signifie que les paramètres à apprendre seraient ajustés pour diminuer la perte associée à une entrée et qu'ils seraient ensuite ajustés pour diminuer la perte associée à l'entrée suivante, ce qui pourrait annuler l'ajustement précédent et augmenter la perte associée à l'entrée précédente. En théorie, l'approche correcte consiste à fournir l'ensemble des données en une seule fois au réseau, en produisant un ensemble de scores pour chaque entrée. À partir de cet ensemble de scores, nous pouvons calculer une perte moyenne pour l'ensemble des données. Dans cette situation, nous saurions comment chaque paramètre à apprendre contribue à cette perte moyenne et nous pourrions appliquer l'algorithme du gradient pour ajuster les paramètres de telle sorte qu'en moyenne la perte sur l'ensemble des données diminue.

Cette approche est cependant trop exigeante en ressources de calcul pour être pratique. Par conséquent, nous utilisons une solution qui se situe entre l'utilisation d'une seule entrée et l'utilisation de la base de données entière. Cette approche est l'algorithme du gradient stochastique et consiste à diviser l'ensemble de données en sous-ensembles appelés lots et à appliquer chaque lot séparément au réseau. La taille des lots est un autre paramètre qui, tout comme le taux d'apprentissage, doit être défini par l'utilisateur avant le début de l'apprentissage. Cette approche peut être considérée comme une approximation stochastique de l'optimisation par descente de gradient, puisqu'elle remplace le gradient réel (calculé à partir de l'ensemble des données) par une estimation de celui-ci (calculée à partir d'un sous-ensemble de données choisi au hasard). En particulier dans les problèmes d'optimisation à haute dimension, cela permet de réduire la charge de calcul, en obtenant des itérations plus rapides en échange d'un taux de convergence plus faible.

Des lots de données sont appliqués au réseau et après chaque lot, l'algorithme du gradient est utilisé avec l'algorithme de rétropropagation pour mettre à jour les paramètres à apprendre du réseau. Après le passage de tous les lots dans le réseau, une époque est passée. L'ensemble du processus est répété pour autant d'époques que l'on souhaite.

## Optimiseurs

Il existe différentes façons de mettre à jour les paramètres du modèle autres que l'algorithme du gradient. La technique spécifique qui est utilisée pour mettre à jour les paramètres est connue sous le nom d'optimiseur. La plupart des optimiseurs s'appuient sur l'algorithme du gradient et résolvent certains problèmes de cette approche. Une version légèrement modifiée de l'algorithme du gradient stochastique introduit l'idée de momentum, qui ajoute à la direction actuelle de mise à jour des paramètres une fraction de la direction précédente, créant ainsi cette idée de momentum. Ceci est utile pour échapper aux minimums locaux.



## **Surapprentissage**

A la fin de plusieurs époques, il est presque garanti que le réseau aura une bonne performance dans la prédiction des classes de données qui ont été utilisées pour l'apprentissage. Cela ne signifie pas, cependant, que le réseau aura une bonne performance dans la prédiction de la classe des données complètement nouvelles. Les classes des données d'apprentissage sont déjà connues, notre objectif est que l'algorithme apprenne à partir des données d'apprentissage et soit capable de classer correctement des données complètement nouvelles. Le phénomène selon lequel un réseau a de bonnes performances en matière de données d'apprentissage et de mauvaises performances avec les nouvelles données est connu sous le nom de surapprentissage et constitue un grand obstacle dans le machine learning.

Le surapprentissage est la production d'une analyse qui correspond trop précisément à un ensemble particulier de données, et qui peut donc ne pas correspondre à des données supplémentaires ou ne pas prévoir de manière fiable les observations futures. Si un réseau a trop de paramètres, il peut développer une correspondance de type dictionnaire entre les données d'apprentissage et la sortie correspondante, ce qui est contre-productif lorsqu'il s'agit de données inédites.

## **Évaluation du réseau**

Afin d'avoir une idée des performances du réseau sur les nouvelles données et de déterminer s'il y a un surapprentissage, l'ensemble de données initial est divisé en deux sous-ensembles, un ensemble d'apprentissage et un ensemble de test. L'ensemble d'apprentissage est utilisé pour l'apprentissage du réseau, comme expliqué précédemment, et l'ensemble de test ne l'est pas. À la fin de chaque époque, la performance du réseau est mesurée sur l'ensemble d'apprentissage et sur l'ensemble de test. En comparant ces deux résultats, nous pouvons avoir une idée de l'existence éventuelle d'un surapprentissage et de l'époque à laquelle ce surapprentissage commence. Différentes mesures sont utilisées pour évaluer la performance du réseau. Parmi les plus courantes, on trouve des graphiques de l'exactitude par époque et de la perte par époque. En analysant ces graphiques, nous pouvons déterminer l'existence d'un surapprentissage s'il y a une grande différence entre le graphique concernant les données d'apprentissage et le graphique concernant les données de test. Nous pouvons également, par exemple, déterminer le moment où le surapprentissage commence à l'époque où la perte de test commence à augmenter alors que la perte d'apprentissage continue à diminuer.

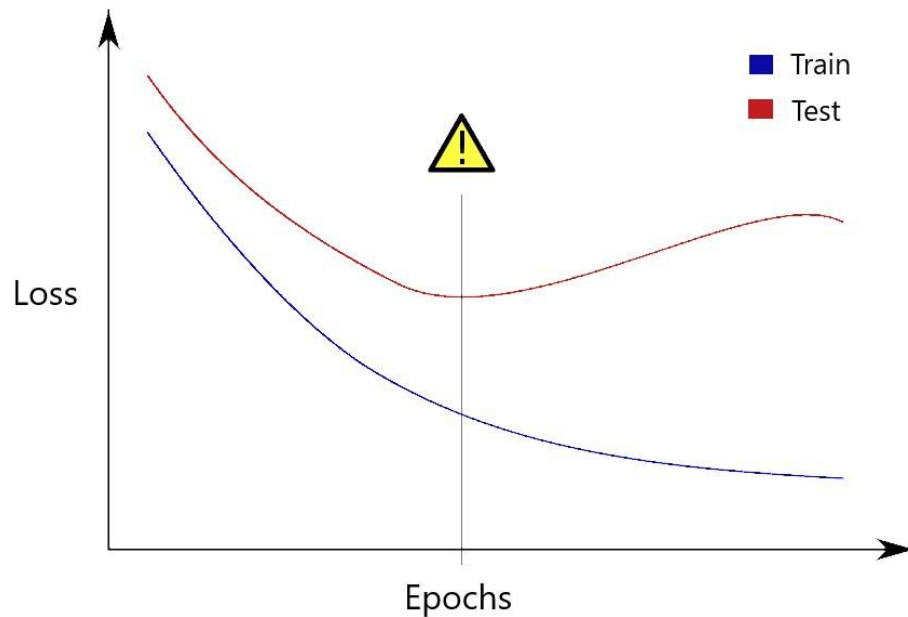


Fig 24 - Exemple de graphique de perte d'apprentissage et de test

## Hyperparamètres

Un hyperparamètre est un paramètre du réseau qui n'est pas appris automatiquement mais qui doit être défini par l'utilisateur avant le début de l'apprentissage. Des exemples d'hyperparamètres sont le taux d'apprentissage, la taille du lot, la largeur des couches, la profondeur du réseau, etc.

## Le réglage des hyperparamètres et l'ensemble de validation

Lors de l'expérimentation des réseaux de neurones, nous testons souvent différentes valeurs pour les hyperparamètres afin de découvrir quelles valeurs produisent les meilleurs résultats, c'est ce qu'on appelle le réglage des hyperparamètres. Cette performance est évaluée sur un ensemble de tests qui contient des données jamais vues antérieurement. Cependant, ces données inédites ne sont qu'un sous-ensemble de toutes les données inédites. Cela signifie qu'il est possible de surapprendre l'ensemble de test et de choisir des hyperparamètres qui augmentent les performances sur l'ensemble de test, mais pas nécessairement sur l'ensemble complet des données inédites. L'utilisation des performances du réseau sur l'ensemble de test pour régler les hyperparamètres entraîne un phénomène appelé fuite de données. Le réseau n'apprend pas directement de l'ensemble de test comme il le fait pour l'ensemble d'apprentissage, mais les particularités de l'ensemble de test se répercutent sur le réseau lorsque nous choisissons l'hyperparamètre pour augmenter les performances de l'ensemble de test. Cela signifie que si nous utilisons l'ensemble de test pour le réglage des hyperparamètres, il cesse d'être un ensemble de données véritablement inédites. C'est pourquoi l'ensemble de données d'origine est souvent divisé en trois sous-ensembles : apprentissage, validation et test. L'ensemble d'apprentissage est utilisé pour l'apprentissage du réseau et l'ensemble de validation est utilisé comme un ensemble de données inédites pour le réglage des hyperparamètres. Une fois le

réglage des hyperparamètres terminé, l'ensemble d'apprentissage et l'ensemble de validation sont combinés pour l'apprentissage et l'ensemble de test est utilisé une seule fois comme un ensemble de données véritablement inédites pour vérifier la performance finale du réseau. Le réseau ne doit pas être modifié en fonction des performances de l'ensemble de test, car cela entraînerait des fuites de données et rendrait les mesures inutiles.

## Annexe 2 - Évolution des mesures du modèle

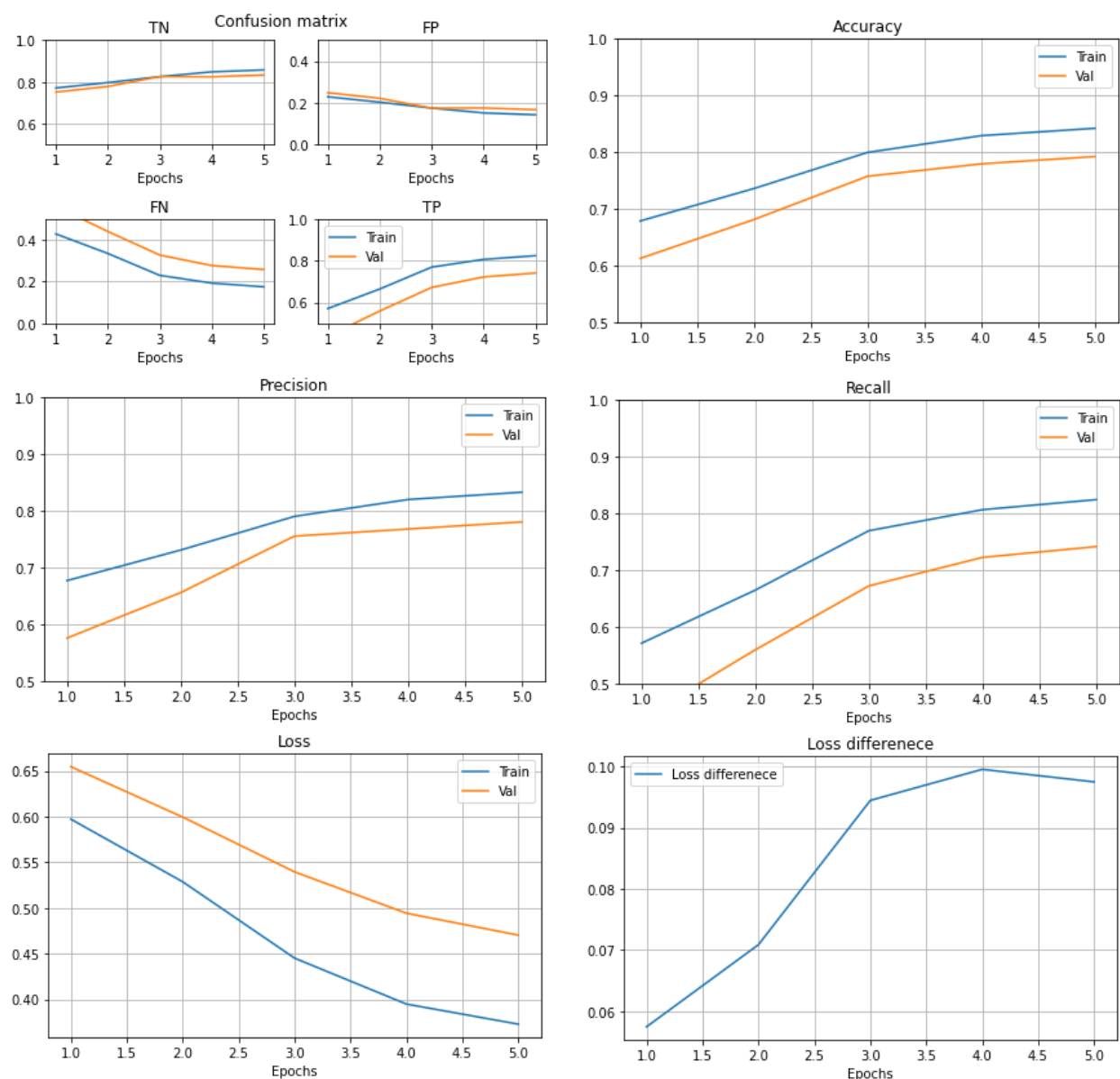


Fig 25 - Évolution des mesures du VGG16

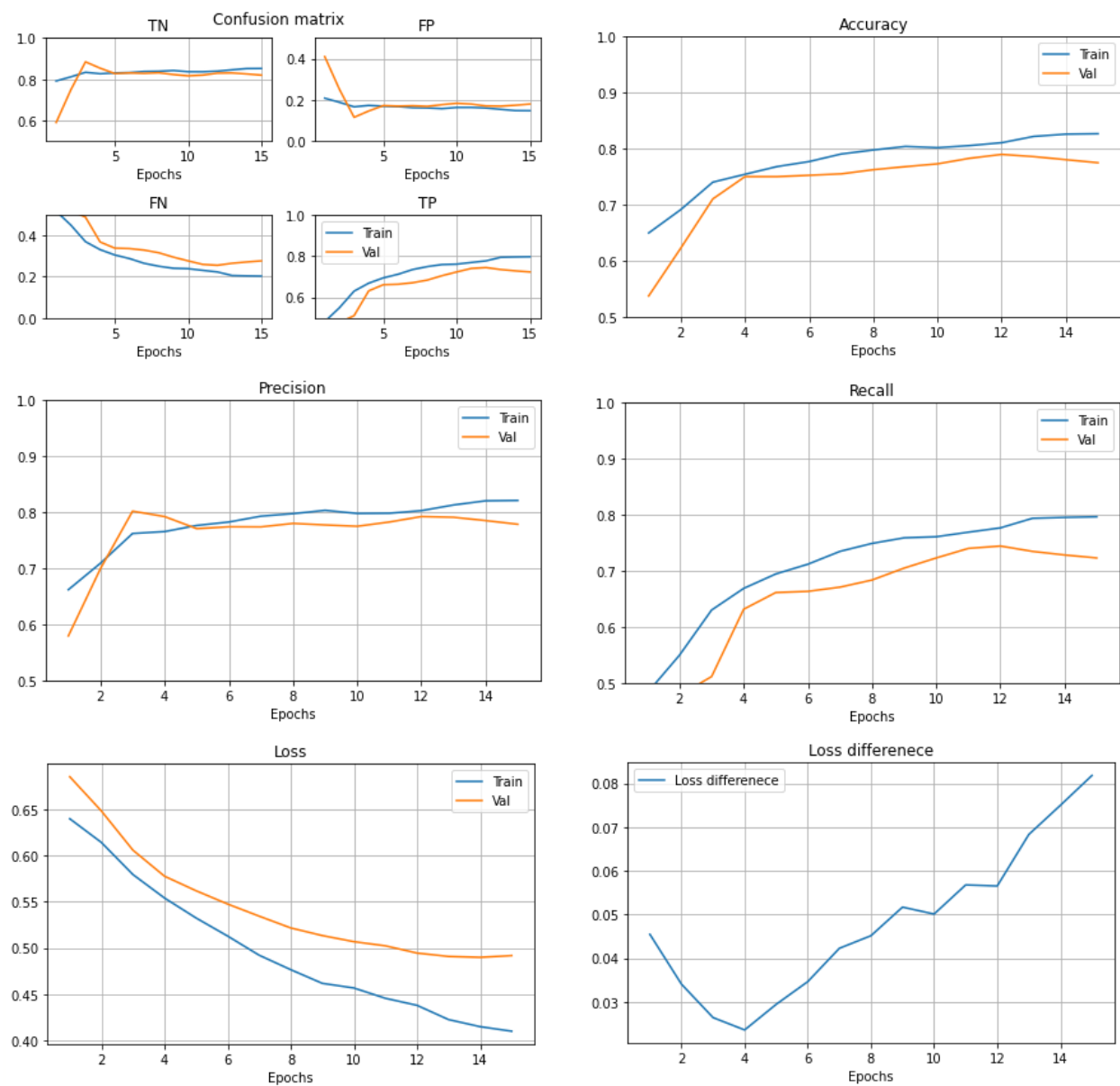


Fig 26 - Évolution des mesures du ResNet-50

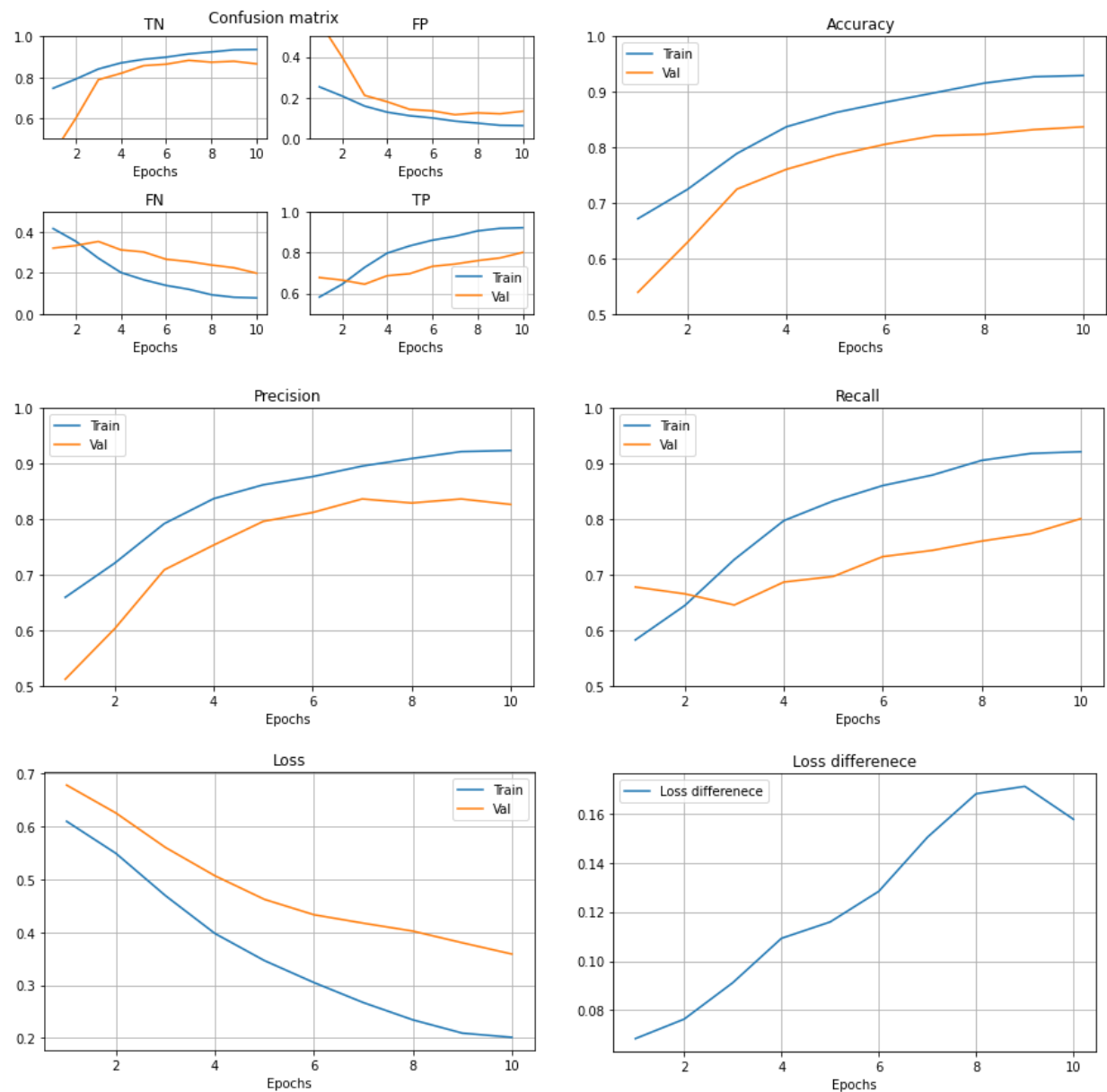


Fig 27 - Évolution des mesures du DenseNet-169

## Bibliographie

- [1] *FDA* [En ligne]. 16 juillet 2020 [Consulté le 10 septembre 2020]. Coronavirus Testing Basics. Disponible sur <https://www.fda.gov/consumers/consumer-updates/coronavirus-testing-basics>.
- [2] Fang et al. Sensitivity of Chest CT for COVID-19: Comparison to RT-PCR. *Radiology*, 2020, Vol. 296, N. 2
- [3] *RadiologyInfo* [En ligne]. 14 février 2018 [Consulté le 10 septembre 2020]. Computed Tomography (CT) - Chest. Disponible sur <https://www.radiologyinfo.org/en/info.cfm?pg=chestct>
- [4] *Radiology Assistant* [En ligne]. [Consulté le 10 septembre 2020]. COVID-19 Imaging findings. Disponible sur <https://radiologyassistant.nl/chest/covid-19/covid19-imaging-findings>
- [5] *Radiopaedia* [En ligne]. Sharma et al [Consulté le 10 septembre 2020]. COVID-19. Disponible sur <https://radiopaedia.org/articles/covid-19-4>
- [6] Ai et al. Correlation of Chest CT and RT-PCR Testing for Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases. *Radiology*, 2020, Vol. 296, N. 2
- [7] *HCPLive* [En ligne]. Sturts, Adam. 21 avril 2020 [Consulté le 10 septembre 2020]. Despite Early Data, CT Alone Not Appropriate for COVID-19 Diagnosis. Disponible sur <https://www.hcplive.com/view/ct-not-appropriate-for-covid19-diagnosis>.
- [8] Zhao, Jinyu, et al. COVID-CT-Dataset: a CT scan dataset about COVID-19. *arXiv preprint arXiv:2003.13865* (2020).
- [9] *European Centre for Disease Prevention and Control* [en ligne]. 25 septembre 2020 [Consulté le 10 septembre 2020]. Q & A on COVID-19: Basic facts. Disponible sur <https://www.ecdc.europa.eu/en/covid-19/facts/questions-answers-basic-facts>
- [10] Simonyan, Karen, et Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [11] Selvaraju, Ramprasaath R., et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE international conference on computer vision*. 2017.
- [12] He, Kaiming, et al. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [13] Ioffe, Sergey, and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).
- [14] Huang, Gao, et al. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- [15] Chollet, Francois. Deep Learning mit Python und Keras: *Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.



- [16] Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin. Learning from data. Vol. 4. New York, NY, USA: *AMLBook*, 2012.
- [17] Murphy, Kevin P. Machine learning: a probabilistic perspective. *MIT press*, 2012.
- [18] Goodfellow, Ian, et al. Deep learning. Vol. 1. Cambridge: *MIT press*, 2016.
- [19] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. *O'Reilly Media*, 2019.