

Apprenant : Maxime Favot

# Projet Chef d'Oeuvre

## *“PneumoniApp”*

*Détection instantanée de cas de pneumonie sur des radios de  
poumons de patients*



## SOMMAIRE

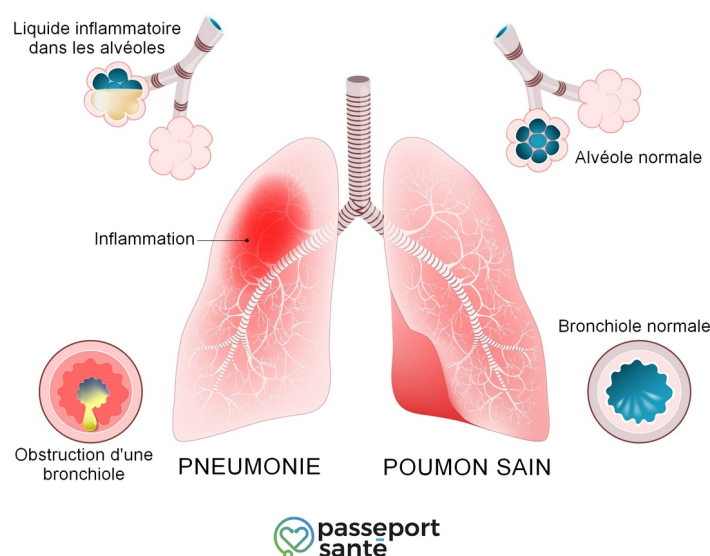
<b>Introduction</b>	<b>3</b>
Contexte	3
Problématique	3
Analyse des besoins	4
<b>Gestion du Projet</b>	<b>5</b>
<b>Intelligence Artificielle</b>	<b>6</b>
Dataset	6
Méthodes choisies	12
Résultat et conclusion	16
<b>Application</b>	<b>17</b>
Base de donnée	18
Backend	21
Qualité et Monitoring	23
Frontend	25
<b>Conclusion</b>	<b>28</b>

# Introduction

## Contexte

La pneumonie se définit comme une infection respiratoire aiguë affectant les poumons. Ceux-ci sont constitués d'alvéoles qui se remplissent d'air quand une personne en bonne santé respire.

En cas de pneumonie, les alvéoles sont remplies de pus et de liquide, ce qui rend la respiration douloureuse et limite l'absorption d'oxygène. La pneumonie est la première cause infectieuse de mortalité chez l'enfant. En 2017, 808 694 d'enfants de moins de 5 ans sont décédés des suites d'une pneumonie, soit 15% des décès dans ce groupe d'âge à l'échelle mondiale.



## Problématique

La pneumonie se manifeste habituellement par une zone opaque sur les clichés radiographiques. En revanche, le diagnostic sur radio est compliqué à cause du nombre d'autres maladies dans les poumons comme l'oedème pulmonaire, l'hémorragie pulmonaire, le cancer des poumons, etc). De plus, à l'extérieur même des poumons, un épanchement pleural apparaît aussi comme une zone opaque sur la radiographie.

Quand cela est possible, une comparaison des clichés radiographiques pris à différents moments, et en corrélation avec les symptômes cliniques et les antécédents familiaux peuvent se révéler utile pour établir un diagnostic.

On se rend facilement compte qu'établir un diagnostic est quelque chose de complexe, qui nécessite une expérience significative chez les médecins.

En raison du très grand nombre de patients touchés par cette infection respiratoire à l'échelle mondiale, et du temps relativement long de diagnostic par les médecins, nous allons à travers ce papier, proposer une solution IA permettant de détecter des cas de pneumonie sur des clichés radiographiques.

La plus value de cette IA serait de permettre un diagnostic quasi instantané et aussi fiable qu'un médecin dans la majorité des cas.

Également, cette solution permettrait de soulager les médecins, et ainsi débloquer du temps supplémentaire pour traiter d'autres patients ou d'autres maladies.

## Analyse des besoins

Afin de réaliser ce projet, nous avons besoin de définir les besoins de chaque intervenant. Côté technique, nous avons besoin de données, en nombre suffisant pour avoir un diagnostic supérieur ou égal à la baseline. On aura besoin de clichés radiographiques labellisés et identifiés pour chaque patient.

Côté praticien, nous allons devoir développer une interface simple qui permet d'uploader une photo à analyser, et avoir en retour un diagnostic, avec son niveau de certitude.

Le modèle sera en mesure de dire si la condition du patient n'est ni "normale" ni en présence de pneumonie, afin de communiquer au médecin qu'il faut pousser un peu plus loin les investigations pour ce cas précis.

# Gestion du Projet

Afin de gérer au mieux ce projet du début à la fin, nous avons décidé de mettre en œuvre une organisation très répandue dans les activités numériques, l'Agilité, et notamment le framework Kanban.

Kanban une méthode agile de gestion de projet visuelle et continue. Cela se traduit par la mise en place dès le jour 1, de différents outils visant à suivre et optimiser la gestion du projet, et en se concentrant sur les tâches les plus importantes en premier lieu.

Dans notre cas, nous avons mis en place un tableau dynamique, réalisé sur le site trello.

La première chose fut de définir le DoD (Definition of Done). Cette étape permet d'établir les conditions qui permettent de valider une tâche. Dans notre cas, le DoD est : "La tâche est réalisée dans son ensemble, validée par l'équipe, et il n'est plus nécessaire de revenir dessus en l'état"

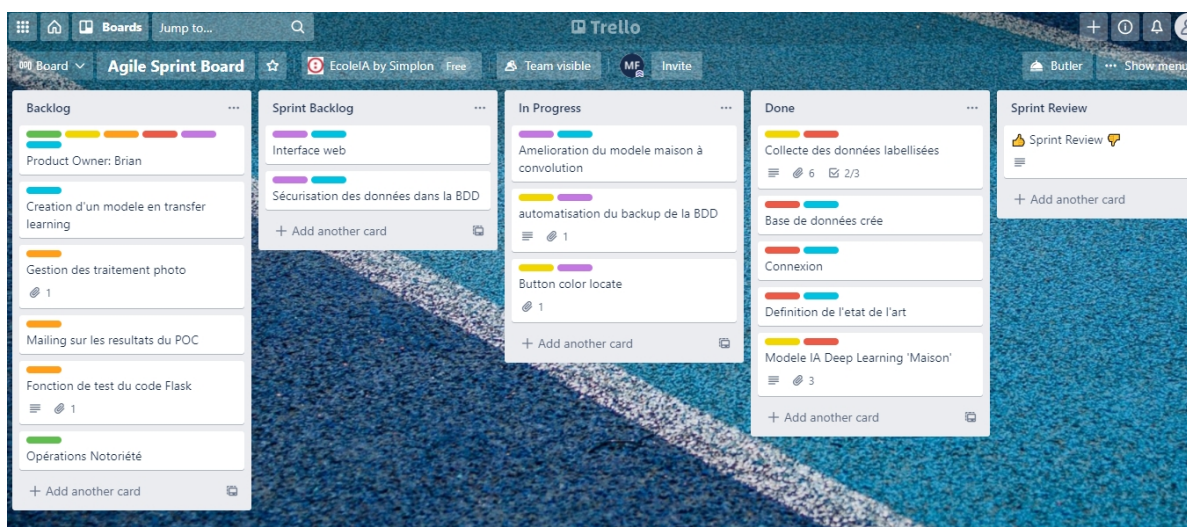
Le tableau se découpe en plusieurs blocs:

**Backlog**, contient toutes les tâches à faire, durant le cycle de vie du projet. Il est bien entendu évolutif dans le temps, ce qui confère une certaine agilité au projet.

**Le sprint backlog**, quant à lui, est un rassemblement de tâches à faire dans un temps défini (ou sprint) et il en résulte le glissement de ces tâches dans les briques suivantes du tableau, au fur et à mesure de l'avancement de ces tâches.

**In progress**, est la où les tâches en cours, le non terminées.

**Done**, les tâches sont validées, et en adéquation avec le Definition of Done.



Il est intéressant de faire après chaque sprint, une sprint review. Elle sert à évaluer et livrer au client la version du produit à cet instant « T ». Il est alors temps de récolter le feedback du client, c'est-à-dire ses retours, positifs comme négatifs, et de les traiter de façon réactive.

Enfin, Une **rétrospective de sprint** permet enfin de revenir sur ce qui a été fait, d'en dégager les points forts et les points faibles, de se remettre en question et de perfectionner la méthodologie de travail pour le sprint suivant.

## Intelligence Artificielle

L'élément central de ce projet réside dans le développement d'un modèle IA permettant de classifier avec précision des clichés radio de poumons, en trois catégories:

**Lung Opacity** => Le cliché se révèle positif à un virus de la pneumonie.

**Normal** => Le patient ne présente pas d'opacité dans les poumons, donc négatif à la pneumonie.

**No Lung Opacity but not Normal** => Le patient ne laisse pas apparaître d'opacités pulmonaires liées à une pneumonie, mais le cliché radio est interprété comme anormal. Cette classe permettra un diagnostic humain dans un second temps.

## Dataset

Nous avons pu récupérer des données nécessaires à l'apprentissage du modèle IA sur le site de la Radiological Society of North America (RSNA).

Cet organisme a lancé un concours de détection de pneumonie en 2018, et a mis à disposition un large dataset labellisé afin de réunir les meilleures conditions pour avoir des résultats utiles à la société, alors que la pneumonie est une des causes majeures de mortalité dans le monde.



Nous avons pu récupérer 26684 fichiers médicaux de type DICOM.

La norme DICOM "**D**igital **I**maging and **C**ommunication in **M**edicine" est un document qui définit une méthode de communication pour les différents équipements d'imagerie médicale numérique. Cette norme est maintenant utilisée par la plupart des fabricants de matériel d'imagerie médicale.

Chacun de ses fichiers DICOM contient un cliché radio, que nous devons extraire, et des métadonnées.

- Le cliché sera extrait du DICOM grâce à la librairie Python Pydicom qui récupérera un array, que nous transformerons en photo grâce à la librairie OpenCV de python également.



- Les métadonnées quant à elles contiennent des informations sur le patient, sur le matériel d'imagerie utilisé et les différents réglages techniques pour la prise du cliché.

```
[18] ▶ MI
#on lit les metadata du fichier dcm grace à pydicom
pydicom.read_file(dcm_file)

Dataset.file_meta -----
(0002, 0000) File Meta Information Group Length  UL: 202
(0002, 0001) File Meta Information Version        OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: Secondary Capture Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0002, 0010) Transfer Syntax UID                 UI: JPEG Baseline (Process 1)
(0002, 0012) Implementation Class UID           UI: 1.2.276.0.7230010.3.0.3.6.0
(0002, 0013) Implementation Version Name        SH: 'OFFIS_DCMTK_360'
-----
(0008, 0005) Specific Character Set               CS: 'ISO_IR 100'
(0008, 0016) SOP Class UID                       UI: Secondary Capture Image Storage
(0008, 0018) SOP Instance UID                    UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0008, 0020) Study Date                          DA: '19010101'
(0008, 0030) Study Time                          TM: '000000.00'
(0008, 0050) Accession Number                    SH: ''
(0008, 0060) Modality                            CS: 'CR'
(0008, 0064) Conversion Type                     CS: 'WSD'
(0008, 0090) Referring Physician's Name          PN: ''
(0008, 103e) Series Description                   LO: 'view: PA'
(0010, 0010) Patient's Name                      PN: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0020) Patient ID                          LO: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0030) Patient's Birth Date                DA: ''
(0010, 0040) Patient's Sex                       CS: 'F'
(0010, 1010) Patient's Age                       AS: '51'
(0018, 0015) Body Part Examined                  CS: 'CHEST'
(0018, 5101) View Position                       CS: 'PA'
(0020, 000d) Study Instance UID                   UI: 1.2.276.0.7230010.3.1.2.8323329.28530.1517874485.775525
(0020, 000e) Series Instance UID                  UI: 1.2.276.0.7230010.3.1.3.8323329.28530.1517874485.775524
(0020, 0010) Study ID                            SH: ''
(0020, 0011) Series Number                       IS: '1'
(0020, 0013) Instance Number                     IS: '1'
(0020, 0020) Patient Orientation                  CS: ''
(0028, 0002) Samples per Pixel                    US: 1
(0028, 0004) Photometric Interpretation          CS: 'MONOCHROME2'
-----
```

Il peut être intéressant de coupler métadonnées et clichés radio pour entraîner un modèle IA dans certains cas.

Dans notre cas de figure, et car il s'agit de données médicales soumises à la RGPD (Règlement Général sur la Protection des Données), nous n'avons pas vraiment d'informations probantes pouvant permettre au modèle de mieux classer les radios pulmonaires, hormis l'âge et le genre du patient.

Nous avons pris le parti de ne pas faire d'entraînement parallèle du modèle IA et de se concentrer sur les clichés radio uniquement, d'autant plus que le dataset est relativement fourni.

Concernant la labellisation, le site de la RSNA met à disposition un fichier .csv contenant le diagnostic associé à chaque fichier DICOM du Dataset. Le lien entre le label et le cliché se fait par l'identification du patient. 'Id\_Patient'

Il est à noter que plusieurs clichés peuvent être associés au même identifiant patient.

```
▶ MI
# Aperçu de la table df_Cible
df_Cible.head()
```

	patientId	classe
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	No Lung Opacity / Not Normal
1	00313ee0-9aaa-42f4-b0ab-c148ed3241cd	No Lung Opacity / Not Normal
2	00322d4d-1c29-4943-afc9-b6754be640eb	No Lung Opacity / Not Normal
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	Normal
4	00436515-870c-4b36-a041-de91049b9ab4	Lung Opacity



Avant de commencer, il est important d'explorer les données: Nous allons en premier lieu supprimer les valeurs vides, et vérifier la distribution des outputs.

```
▶ ▶ M↓
```

```
# On verifie si des valeurs NAN sont présentes dans le dataset  
df_cible.isnull().values.any()
```

False

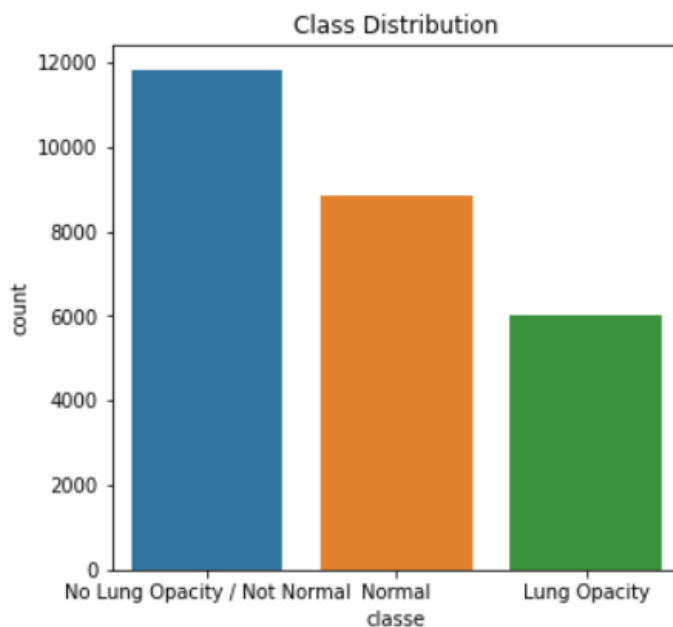
## Data Visualization

```
[10] ▶ ▶ M↓
```

```
fig = plt.figure(figsize=(18,5))  
fig.add_subplot(1, 3, 1)  
ax = sb.countplot(df_full['classe'])  
ax.set(title = 'Class Distribution')  
plt.show()
```

c:\users\maxime\desktop\pneumoniaApp\pneumo\110\site-packages\seaborn\\_decorators.py:109: FutureWarning: The `order` argument to `countplot` is deprecated. From version 0.12, the only valid positional argument will be `x`. All other positional arguments

FutureWarning



```
[11] ▶ MI
#https://www.python-course.eu/python3_formatted_output.php
class_disc = df_full['classe'].value_counts()
print('Pourcentage de patients avec No Lung opacity/Not Normal : {:5d} or {:.2f}%'.format(class_disc[0],(class_disc[0]/df_full['classe'].count())*100))
print('Pourcentage de patients avec Lung opacity : {:5d} or {:.2f}%'.format(class_disc[1],(class_disc[1]/df_full['classe'].count())*100))
print('Pourcentage de patients avec Normal : {:5d} or {:.2f}%'.format(class_disc[2],(class_disc[2]/df_full['classe'].count())*100))

Pourcentage de patients avec No Lung opacity/Not Normal : 11821 or 44.30%
Pourcentage de patients avec Lung opacity : 8851 or 33.17%
Pourcentage de patients avec Normal : 6012 or 22.53%
```

On remarque que les classes sont relativement équilibrées, nous n'avons pas de traitements de rééquilibrage à effectuer.

Maintenant il s'agit de créer une fonction permettant formater les données afin qu'elles soient utilisées par le modèle. La fonction suivante est utilisée:


```
[23] ▶ MI
# Fonction qui permet de mettre dans un dictionnaire le chemin de l'image, le label (classe), et l'index
def extract_data(dataset):
    datacol = {}
    index = 0
    for n, row in dataset.iterrows():
        pid = row['patientId']
        if pid not in datacol:
            index = index+1
            datacol[pid] = {
                'dicom': train_images_dir + '/%s.dcm' % pid,
                'label': row['classe'],
                'index': index}
    return datacol
```

Enfin, on utilise une fonction pour créer nos données utilisables pour notre futur modèle de deep learning.

```
def generateData(n):
    Y_list = []
    ARRAY_DIM = int(n)
    X = np.zeros((ARRAY_DIM, IMAGE_DIM, IMAGE_DIM, 3))
    X.shape
    for i, (key, value) in enumerate(list(bbox_data_dict.items())[0:n]):
        print(value['dicom'], value['label'], value['index'])
        dcm_path = value['dicom']
        index = value['index']
        target = value['label']
        dcm_data = pydicom.read_file(dcm_path)
        img = dcm_data.pixel_array
        img = cv2.resize(img, dsize=(IMAGE_DIM, IMAGE_DIM), interpolation=cv2.INTER_CUBIC)
        img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
        X[i] = img
        Y_list.append(value['label'])
    Y = pd.get_dummies(pd.DataFrame(data=Y_list, columns=['class']))
    return X, Y
```

Il ne reste plus qu'à utiliser la fonction `train_test_split` de `sklearn` pour définir nos données d'entraînement, et de validation. Ici nous avons choisi un ratio de 0.2, ce qui est un plutôt courant.

```
from sklearn.model_selection import train_test_split
# Create the Validation Dataset
X_train, X_valid, Y_train, Y_valid = train_test_split(X, Y, train_size=0.8, test_size=0.2, random_state=42)
```


▶  MI

```
X_train.shape, X_valid.shape, Y_train.shape, Y_valid.shape
```

(4269, 256, 256, 3), (1068, 256, 256, 3), (4269, 3), (1068, 3))

## Data normalisation

Normalisation est un procédé classique dans le processing d'image qui permet de changer la valeur de l'intensité des pixels. Cette conversion permet d'avoir des valeurs de pixels cohérents entre eux et ainsi normalisés. C'est une étape importante qui facilite l'apprentissage du modèle CNN.

```
▶  MI
```

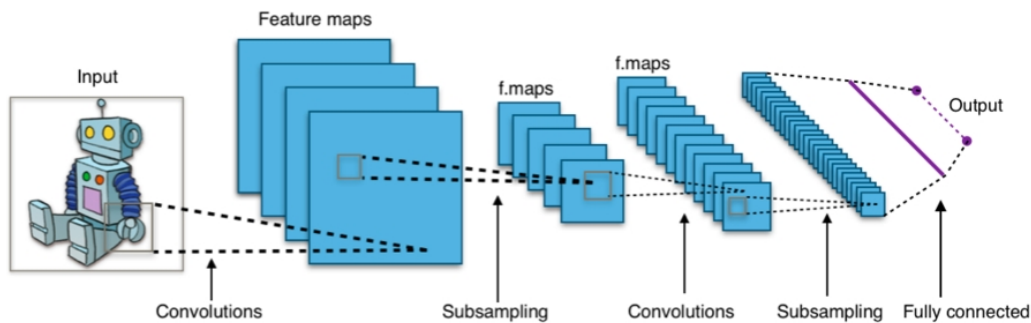
```
# Parse numbers as floats
X_train = X_train.astype('float32')
X_valid = X_valid.astype('float32')

# Normalize input data
X_train = X_train / 255
X_valid = X_valid / 255
```

## Méthodes choisies

Dans un premier temps, le choix s'est porté sur un modèle IA à convolution (CNN), avec un réseau de neurones connectés.

En effet, c'est un modèle utilisé pour classifier des images, et qui est le plus adapté pour ce projet.



Pour cela nous avons testé deux architectures différentes que nous allons voir en détail, puis en comparer les résultats.

#### A) Le CNN

Dans un premier temps, nous avons utilisé le modèle à convolution "classique". Ce modèle prend en entrée une image, et parcourt les différentes couches du réseau. Le but de chaque couche est d'extraire des caractéristiques qui représentent l'image d'entrée dans un niveau sémantique de plus en plus haut. En sortie, le réseau est capable de classifier l'image en plusieurs catégories (ici 3 catégories).

```
model = Sequential([
    Conv2D(8, (3,3), input_shape=(IMAGE_DIM, IMAGE_DIM, 3), activation='relu'),
    MaxPooling2D(2,2),
    Dropout(0.4),

    Conv2D(32, (3,3), activation='relu', kernel_regularizer=regularizers.l2(l=0.01)),
    MaxPooling2D(2,2),
    Dropout(0.4),

    Conv2D(10, (3,3), activation='relu', kernel_regularizer=regularizers.l2(l=0.01)),
    MaxPooling2D(2,2),
    Dropout(0.2),

    Flatten(),
    Dense(3, activation='softmax'),

])
```

Comme on peut le voir, l'architecture est la suivante:

- 1) Une convolution de 8 filtres en 3×3 suivie d'une couche d'activation ReLU (*signifie Rectified Linear Unit, qui veut dire unité linéaire de rectification. Autrement dit, c'est une fonction mathématique qui annule tout ce qui est inférieur à une certaine valeur et conserve tout ce qui est supérieur*)  
Un max-pooling de 2×2,  
Un dropout de 0.4 (Cela signifie que 40% des neurones seront ignorés, on écarte ici un peu plus les risques d'overfitting)
- 2) Une convolution de 32 filtres en 3×3 suivie d'une couche d'activation ReLU,  
Un max-pooling de 2×2  
Un dropout de 0.4
- 3) Une convolution de 10 filtres en 3×3 suivie d'une couche d'activation ReLU,  
Un max-pooling de 2×2  
Un dropout de 0.2
- 4) Un flatten qui va créer le vecteur final à envoyer au réseau de neurones artificiels (ou dense). Il ne prend aucun paramètre, car il n'y a besoin de rien de particulier pour mettre toutes les images bout à bout.
- 5) Un dense, ou réseau de neurones artificiels qui aura 3 neurones (pour 3 catégories à classifier) et sera suivi d'un softmax. Softmax est une fonction mathématique qui permet de normaliser un vecteur pour en faire des probabilités

Enfin, on va compiler le modèle, en lui passant des paramètres de LOSS, Metrics et Optimizer:

```
# Compile the model.  
learning_rate = 0.01  
model.compile(  
    loss='categorical_crossentropy',  
    metrics=['accuracy'],  
    optimizer=Adam(lr=learning_rate))
```

le **loss** : c'est une fonction qui va servir à mesurer l'écart entre les prédictions de notre IA et les résultats attendus. Elle évalue donc la justesse du CNN et permet de mieux l'adapter aux données si besoin. Nous allons utiliser "categorical\_crossentropy" comme loss, car on a des données de type "catégories" en sortie de l'algorithme.

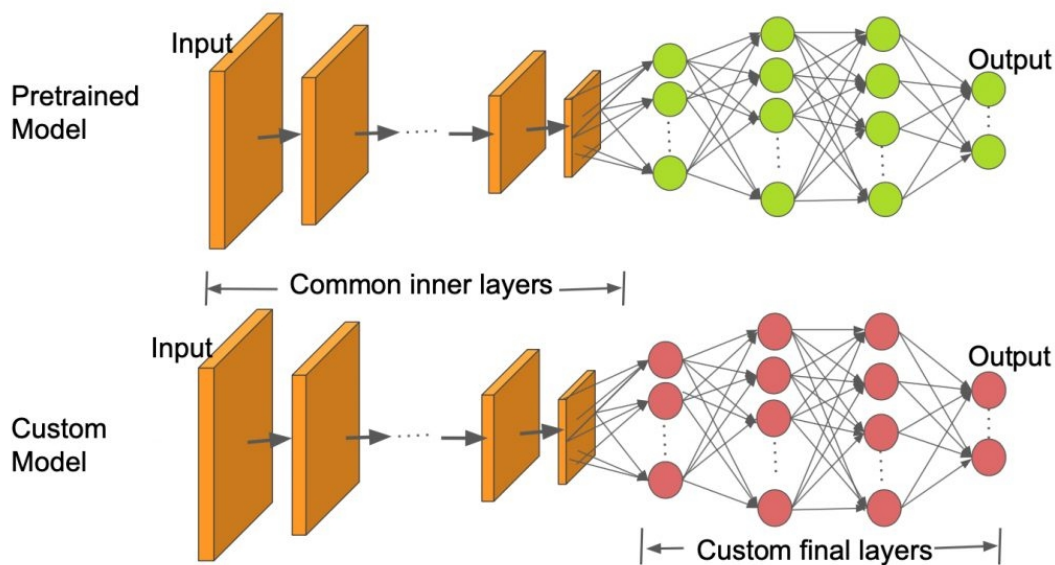
l'**optimizer** : c'est un algorithme qui va dicter comment mettre à jour le CNN pour diminuer le loss, et avoir donc de meilleures prédictions. Ici on s'appuiera sur "adam" (adaptive moment estimation), très souvent utilisé.

La **metrics** : On a ici le même principe que le loss, sauf que la metrics n'est pas utilisée par le CNN, à l'inverse du loss qui sert pour la mise à jour des variables du CNN via l'optimizer. On utilisera cette fois "accuracy", sans que cela ait de réelle importance pour nous.

## B) Transfer Learning

Dans un second temps, nous avons utilisé la méthode du transfer learning pour entraîner notre deuxième modèle. En effet, le deep learning implique une puissance de calcul souvent massive pour fonctionner correctement. Grâce à cette méthode, il est possible d'utiliser des modèles pré-entraînés sur des millions d'images, et de simplement geler les poids des couches d'entraînement (convolution, max pooling) et de rajouter notre réseau Dense qui sera lui spécifique à notre projet de classification des radios de poumons.

Dans le schéma ci-dessous, on a représenté en orange les couches d'entraînement dite "communes", dont on aura gelé les poids. Cette étape permet d'extraire les features les plus générales des photos, alors que le réseau fully connected Dense permet de faire la classification sur une tâche précise.



Dans notre cas, nous avons utilisé le modèle MobileNetV2 avec les poids 'imagenet'. Ce modèle est le premier modèle pré entraîné de Tensorflow. Il est extrêmement léger et petit (en termes de ligne de code et de poids des modèles), et très rapide et très efficace en termes de précision.

```

▶ ▶ MI
from tensorflow.keras.applications.mobilenet import MobileNet
|
IMG_SHAPE = (IMAGE_DIM, IMAGE_DIM, 3)

unfrozen_model = tf.keras.applications.MobileNetV2(
    input_shape=IMG_SHAPE,
    include_top=False,
    weights='imagenet',
)

```

Choisir `include_top = False` indique que l'on veut définir notre propre couche de classification. On indique également que l'on souhaite récupérer les poids entraînés issus d'imagenet.

Enfin, on définit le modèle Sequential en y ajoutant les couches de pooling et Dense pour former le modèle de classification de radio pulmonaires.

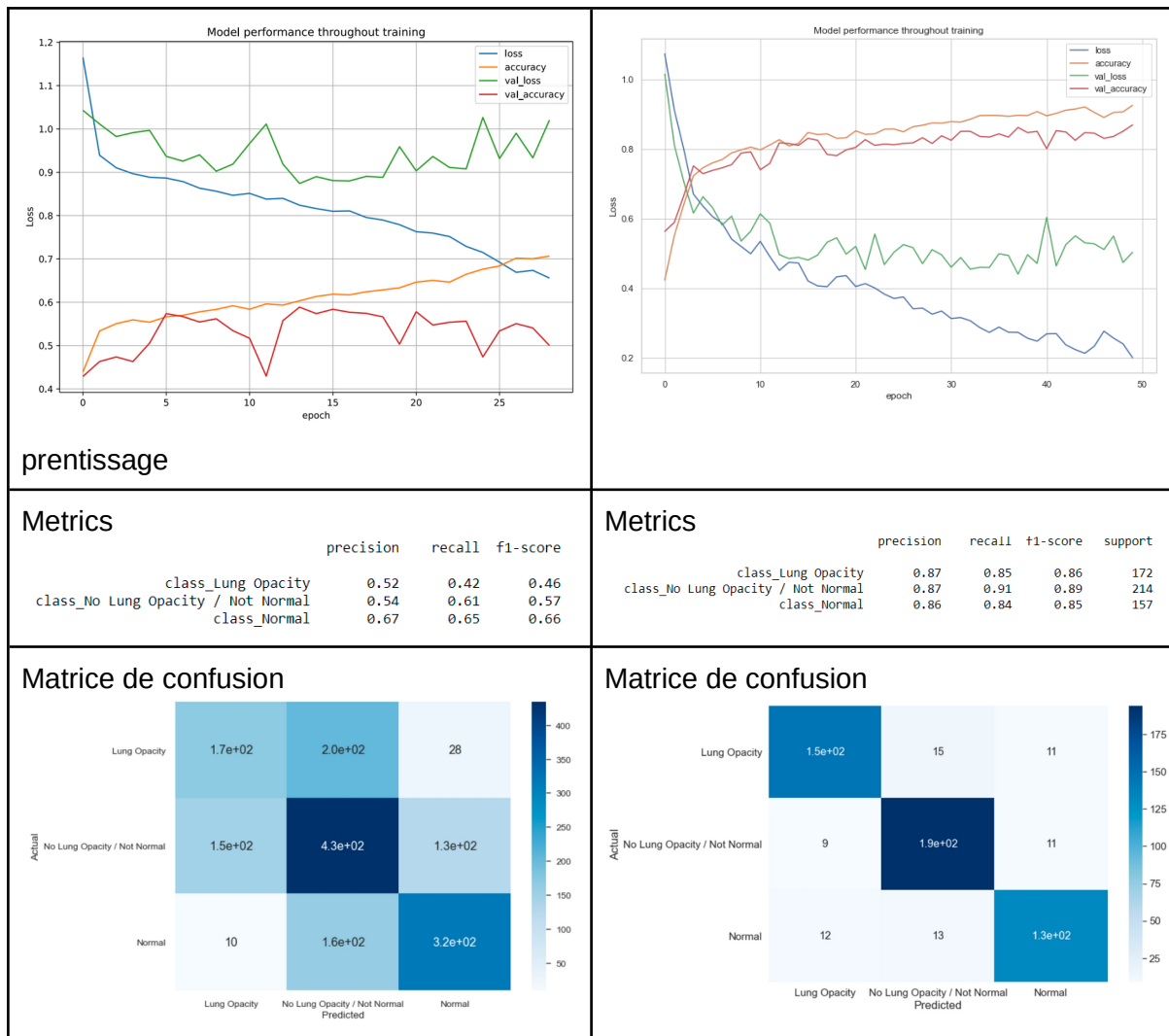
```
model_unfrozen_pretrained = tf.keras.Sequential([
    unfrozen_model,
    global_average_layer,
    prediction_layer
])
```

Après le fit du modèle, l'apprentissage commence et on remarque déjà des différences avec le modèle à convolution 'traditionnel'

## Résultat et conclusion

Nous allons faire un comparatif des résultats des deux méthodes utilisées.

CNN "Classique"	Transfert Learning
Courbe d'ap	Courbe d'apprentissage



On remarque clairement que le modèle issu de transfer learning est bien plus intéressant dans notre projet. En plus d'être rapide et léger, il démontre une capacité de généralisation intéressante, avec une accuracy moyenne de 86 %, en évitant toutefois l'overfitting (malgré la complexité de la structure et le nombre important de paramètres.)

En terme d'amélioration, il serait intéressant d'entraîner un modèle pré entraînés avec d'autres modèles existant (RESNET, Inception, VGG, etc), et d'améliorer la partie fully connected de ces modèles afin de coller au mieux à notre sujet, et ainsi améliorer ses performances.

En ce qui concerne le modèle à convolution classique, on remarque que sur la courbe d'apprentissage, l'accuracy du set de validation ne suit pas la trajectoire de l'accuracy du train set. C'est un signe d'overfitting. Malgré avoir ajouté des dropout et des regularizer L2, et simplifié au maximum la complexité de l'architecture, le résultat reste similaire, avec un gros overfitting et une accuracy légèrement supérieure à 50%. Ce modèle n'est donc pas viable pour la suite de notre projet.

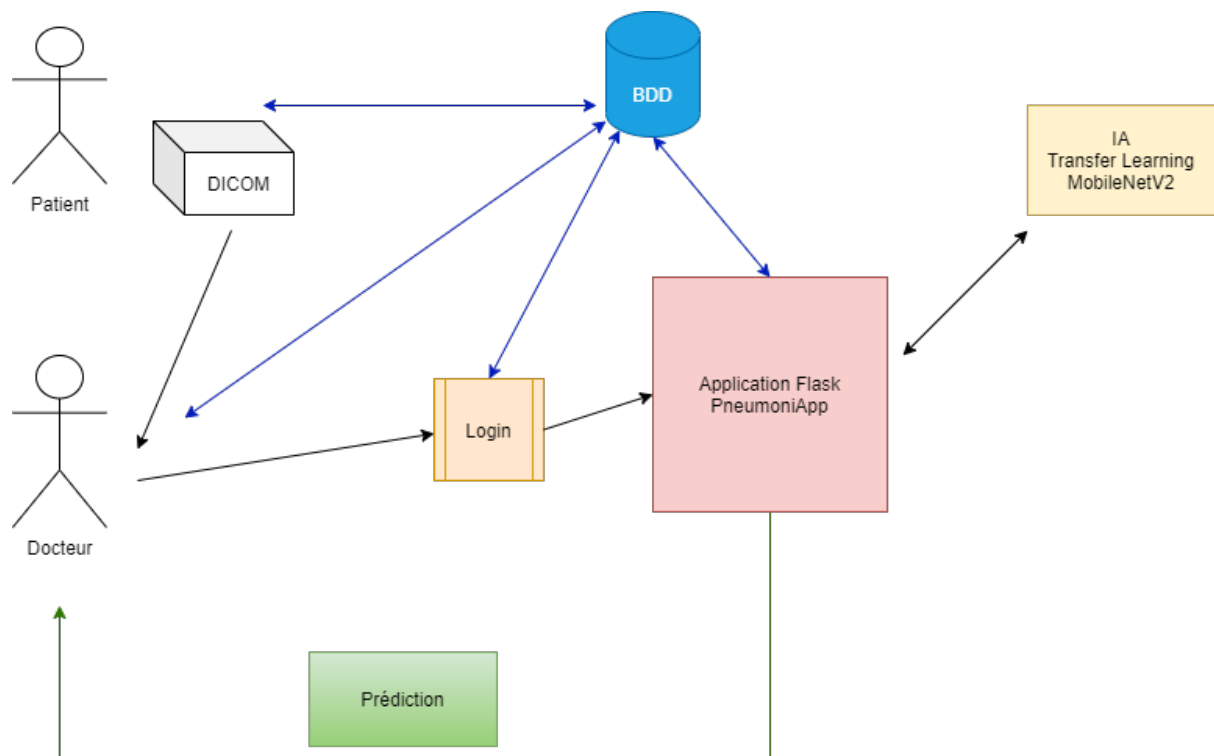


# Application

Le but de l'application PneumoniApp est de faire l'interface entre l'IA et l'utilisateur, ici le médecin. Très simplement, le médecin doit pouvoir être en mesure d'uploader un fichier DICOM de son patient dans l'application, et avoir instantanément une classification de l'état du patient au moment de la prise du cliché.

Afin de sécuriser les données et le rapport Médecin/Patient, l'accès à l'application est disponible uniquement sur inscription, avec renseignement des identifiants du médecin (matricule officiel).

Cette relation Médecin/Patient permet aussi à l'application de proposer un historique des prédictions. Le médecin peut ainsi voir l'évolution de l'état du patient à travers le temps.



## Base de donnée

Nous allons maintenant développer la partie Base de données de notre projet. Nous avons vu que notre projet implique des relations entre plusieurs parties. C'est pourquoi, nous allons créer notre base de données avec Mysql, qui gère les relations entre différentes tables.

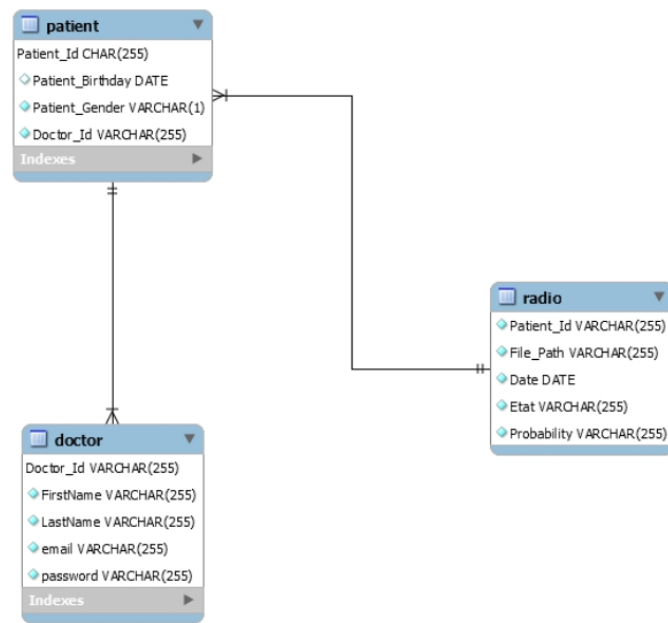
Développé par Oracle, MySQL Workbench est un logiciel qui permet de gérer et d'administrer des bases de données MySQL, et ce, via une interface graphique, c'est pourquoi nous avons utilisé ce logiciel pour notre projet.



Afin de modéliser notre base de données, nous avons créé les tables avec les différentes relations dans Workbench. En voici un exemple :

```
1 • CREATE TABLE `Doctor` (  
2     `Doctor_Id` varchar(255) NOT NULL,  
3     `FirstName` varchar(255) NOT NULL,  
4     `LastName` varchar(255) NOT NULL,  
5     `email` varchar(255) NOT NULL,  
6     `password` varchar(255) NOT NULL,  
7     PRIMARY KEY (`Doctor_Id`)  
8 );  
9  
10 • CREATE TABLE `Patient` (  
11     `Patient_Id` varchar(255) NOT NULL,  
12     `Patient_Birthday` DATE NOT NULL,  
13     `Patient_Gender` varchar(1) NOT NULL,  
14     `Doctor_Id` int(8),  
15     PRIMARY KEY (`Patient_Id`)  
16 );  
17
```

Une fois les tables créées, on peut, grâce à l'outil de reverse engineering, obtenir un schéma de notre base de donnée, avec les relations:



Visuellement, on voit que la table “patient” est liée avec la table “radio” via la clé “Patient\_Id”. De plus, la clé “Doctor\_Id” relie les tables “patient” et “doctor”

Pour peupler la BDD, cela se passe dans l’application, dans notre script python. Voici un exemple d’une requête SQL permettant d’insérer des données dans la table “doctor” suite à l’inscription d’un médecin sur le portail d’identification:

```

@app.route('/register', methods = ['GET', 'POST'])
def register():
    if request.method == 'POST' and 'prénom' in request.form and 'nom' in request.form and 'email' in request.form and 'password' in request.form:
        matricule = request.form['matricule']
        prénom = request.form['prénom']
        nom = request.form['nom']
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM doctor WHERE email=% s', (email, ))
        account = cursor.fetchone()
        if account:
            flash('Cette adresse email est déjà utilisée !', 'error')
        elif not re.match(r'^@[^@]+\.[^@]+$', email):
            flash('Adresse email invalide !', 'error')
        elif not re.match(r'[A-Za-z]+$', prénom):
            flash('Votre prénom doit contenir seulement des lettres !', 'error')
        elif not re.match(r'[A-Za-z]+$', nom):
            flash('Votre nom doit contenir seulement des lettres !', 'error')
        else:
            cursor.execute('INSERT INTO doctor (Doctor_Id, FirstName, LastName, email, password) VALUES (% s, % s, % s, % s, md5(% s))', (matricule, prénom, nom, email, password))
            mysql.connection.commit()
            flash('Enregistrement effectué !', 'success')
            return render_template('login.html')
    elif request.method == 'POST':
        flash('Veuillez remplir le formulaire !', 'error')
    return render_template('register.html')
  
```

Une fois que la base de données commence à contenir des informations, il est nécessaire de penser à faire des sauvegardes automatiques de celles-ci.

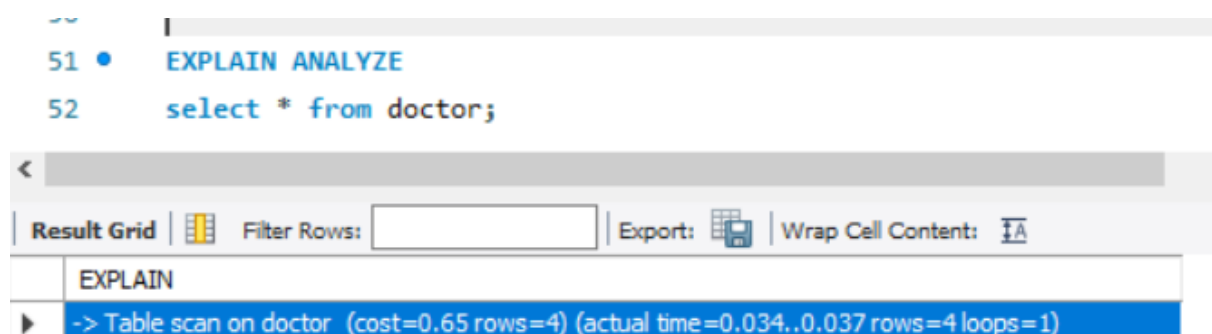
Une méthode pour faire un backup automatisé est d'utiliser le planificateur de tâche de Windows, qui exécutera de façon programmée un petit script .bat qui enregistre physiquement la BDD sur le poste local. Voici un extrait de ce script, et la commande activée par le planificateur:

```
bdd > backup_bdd.bat
1 @echo off
2 Set MYSQL_BIN="C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqldump.exe"
3 Set MYSQL_DB=pneumoniabdd
4 Set MYSQL_USER=root
5 Set MYSQL_PASSWORD=root
6
7 rem date
8 SET DJOUR=%DATE:~0,2%
9 SET DMOIS=%DATE:~3,2%
10 SET DANNEE=%DATE:~6,10%
11 rem heure
12 SET HEURE=%TIME:~0,2%
13 SET MINUTE=%TIME:~3,2%
14 SET SEP=_
15 Set BACKUP_PATH="C:\Users\Maxime\Desktop\PneumoniApp\bdd\"
16
17 SET MYSQL_FILENAME=%BACKUP_PATH%\%MYSQL_DB%%SEP%%DJOUR%%SEP%%DMOIS%%SEP%%DANNEE%%SEP%%HEURE%%SEP%%MINUTE%.sql
18
19 %MYSQL_BIN% -u %MYSQL_USER% -p%MYSQL_PASSWORD% %MYSQL_DB% > %MYSQL_FILENAME%
```

```
(pneumo) PS C:\Users\Maxime\Desktop\PneumoniApp> .\bdd\backup_bdd.bat
```

Pour aller plus loin, il peut être intéressant de mesurer la durée d'exécution d'une requête. En effet, si la BDD devient de plus en plus fournie, cela peut créer des ralentissements d'exécution, qu'il faudra pouvoir améliorer.

Pour cela, on utilise la fonction EXPLAIN ANALYZE avant d'exécuter une requête, et celle-ci nous retourne la durée d'exécution.



The screenshot shows a SQL query execution interface. The query is:

```
51 EXPLAIN ANALYZE
52 select * from doctor;
```

The result grid shows the following output:

EXPLAIN
-> Table scan on doctor (cost=0.65 rows=4) (actual time=0.034..0.037 rows=4 loops=1)

Pour optimiser les temps d'exécution d'une Base de Donnée, il existe notamment la méthode "ACID":

**Atomicité** : Tout ou rien. Soit l'opération se fait en entier, soit elle ne se fait pas du tout. La notion d'atomicité sous-entend la possibilité de défaire une opération avortée.

**Cohérence** : L'opération doit assurer que la base de données sera dans un état valide après l'opération.

**Isolation** : L'opération doit se faire en toute autonomie sans dépendance à une autre opération.

**Durabilité** : En cas de problème important (coupure d'électricité), les modifications apportées sont bien enregistrées.

## Backend

L'application est créée avec Flask, qui est un petit framework web Python léger, qui fournit des outils et des fonctionnalités utiles qui facilitent la création d'applications web en Python.



L'application est constituée d'un script python (app.py) qui permet de faire le lien en la base de donnée Mysql, le Modèle IA pour la prédiction, et les différentes templates de page HTML qui forment l'architecture de l'application.

Voici la partie initialisation de ce script, qui charge le modèle, et se connecte à la base de donnée :

```

app = Flask(__name__)

path='./.env'
load_dotenv(dotenv_path=path)

app.config['MYSQL_USER']=os.getenv("MYSQL_USER")
app.config['MYSQL_PASSWORD']=os.getenv("MYSQL_PASSWORD")
app.config['MYSQL_HOST']=os.getenv("MYSQL_HOST")
app.config['MYSQL_DB']=os.getenv("MYSQL_DATABASE")

SECRET_KEY = os.getenv("SECRET_KEY")
app.config['SECRET_KEY'] = SECRET_KEY
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"

model = tf.keras.models.load_model('./models/modelCNN2.h5')

mysql = MySQL(app)

```

Le script app.py contient aussi les “routes” qui permettent de définir la tâche à exécuter sur la page URL définie. Ici, on décrit la partie prédiction de l'image uploadée, lorsqu'on arrive sur la page “result” de l'application.

```

@app.route('/result', methods=['POST'])
def predict():

    file = request.files['file']
    filename = secure_filename(file.filename)
    filename = os.path.join('./uploads', filename)
    file.save(filename)

    IMAGE_DIM = 256
    dcm = pydicom.read_file(filename)
    img = dcm.pixel_array
    img = cv2.resize(img, dsize=( IMAGE_DIM, IMAGE_DIM), interpolation=cv2.INTER_CUBIC)

    cv2.imwrite('./static/cv2_image.png',img)

    img = cv2.cvtColor(img,cv2.COLOR_GRAY2RGB)
    img = img.reshape(1,IMAGE_DIM, IMAGE_DIM,3)

    prediction = model.predict(img)
    prediction_max = np.max(prediction)
    probability = np.round(prediction_max* 100, 2)
    print(np.argmax(prediction,axis=1))

    proba = "{}%"
    pourcentage = proba.format(probability)

```

Enfin, voici un exemple de page HTML, avec son architecture, ici la page de connexion récupérant les identifiants pour être comparés avec les identifiants valides de la table “doctor” de la base de donnée, pour en valider ou non l'accès au service.

```

<html>
<head>
<meta charset="UTF-8">
<title>Connexion</title>
<link rel="stylesheet" href="{{url_for('static', filename='css/app.css')}}">
</head>
<body>
<div class="element">
<div class="logincontent" align="center">
<div class="logintop">
<h1>Se connecter :</h1>
</div><br>
<div class="loginbottom">
<form action="{{ url_for('login')}}" method="post" autocomplete="off">
{% with messages = get_flashed_messages(with_categories=true) %}
{% for category, message in messages %}
<div class="{{ category }}">{{ message }}</div>
{% endfor %}
{% endwith %}
<br>
<input type="email" name="email" placeholder="Entrez votre adresse email" class="textbox" id="email" required><br><br>
<input type="password" name="password" placeholder="Entrez votre mot de passe" class="textbox" id="password" required><br><br><br>
<input type="submit" class="btn" value="Se connecter">
</form><br></div>
<p class="worddark">Vous êtes nouveau ? <a class="wordlight" href="{{ url_for('register')}}">Créer un compte ici !</a></p>
</div>
</div>
</body>
</html>

```

## Qualité et Monitoring

Lors du développement de l'application, il était indispensable de travailler en mode debug, afin de pouvoir déceler et corriger les erreurs de développement en temps réel. Pour cela, flask permet de définir le mode "debug" :

```

if __name__ == '__main__':
    app.run(debug=True, port=5000)

```

Une fois activé, on lance l'application et nous pouvons travailler en debug. C'est une étape très importante dans le développement de l'application, qui nous a permis de corriger de nombreuses erreurs.

```

2021-07-08 15:15:13.712207341 - tensorflow/compiler/xla/service,
* Debugger is active!
* Debugger PIN: 182-628-386
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

Il est aussi possible de tester le script python en lui-même. Pour cela, il existe une librairie qui permet de faire des tests unitaires: Pytest.

Pour cela, après avoir installé la librairie, on va tester une ou plusieurs fonctions du script comme dans l'exemple ci-dessous.

```
38 def resultat(predicted_class_str):
39     return "Patient is likely to have " + predicted_class_str
40
41
42
43 def test_resultat():
44     assert resultat('Lung Opacity condition') == 'Patient is likely to have Lung Opacity condition'
45
46
```

Pour vérifier la fonction, on lance pytest:

```
(pneumo) PS C:\Users\Maxime\Desktop\PneumoniApp> pytest
===== test session starts =====
platform win32 -- Python 3.7.4, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
rootdir: C:\Users\Maxime\Desktop\PneumoniApp
collected 1 item
```

Si la condition est validée, ici c'est le cas, le résultat de la fonction est bien "Patient is likely to have Lung Opacity condition", alors pytest nous confirme que le test est passé correctement. La fonction est validée.

```
===== 1 passed, 3 warnings in 13.08s =====
(pneumo) PS C:\Users\Maxime\Desktop\PneumoniApp>
```

## Frontend

Pour lancer le script python, on utilise la commande suivante:

```
PS C:\Users\Maxime\Desktop\PneumoniApp> & c:/Users/Maxime/Desktop/PneumoniApp/pneumo/Scripts/Activate.ps1
(pneumo) PS C:\Users\Maxime\Desktop\PneumoniApp> python app.py
```

La page d'accueil se fait sur l'url du localhost : "127.0.0.1 : 5000", et demande au praticien



de se logger. S'il n'a pas créé de compte, il peut le faire en partant de cette page.

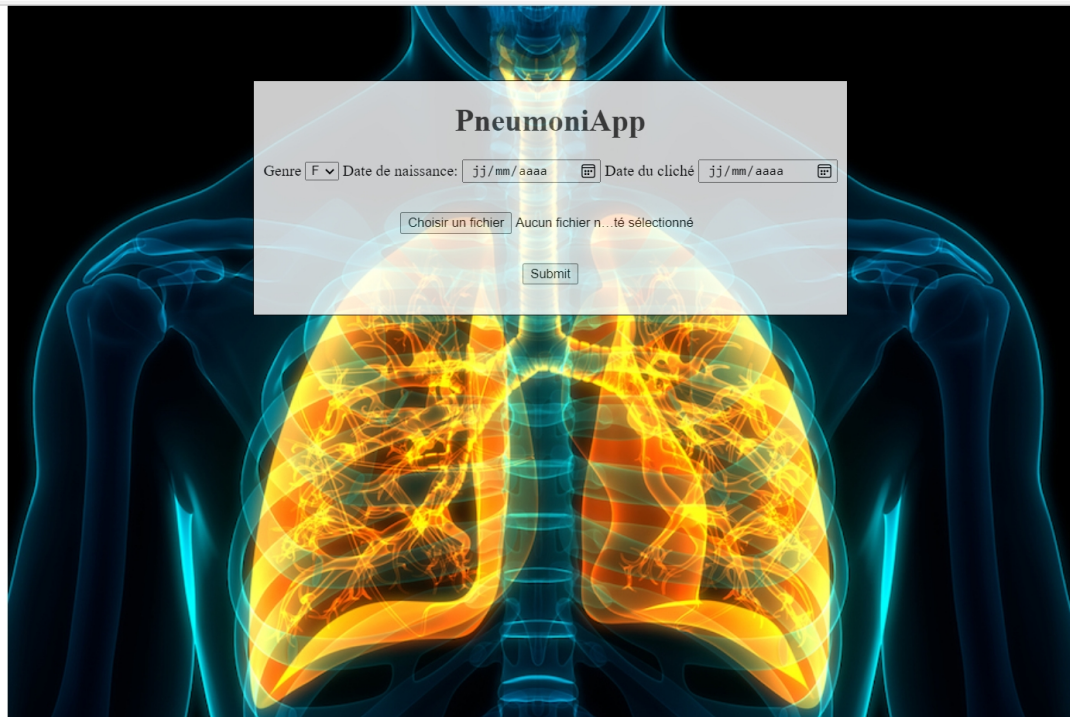
### Se connecter :

Vous êtes nouveau ? [Créez un compte ici !](#)

### Créer un compte

[Connectez vous ici !](#)

Ensuite, une fois loggé, on arrive sur la page "Index" qui permet au praticien de renseigner le genre, la date de naissance et la date du cliché, avant d'uploader le fichier DICOM.



Cette étape lance la prédiction, et permet de peupler la table “patient” de la base de données.

54 • `select * from patient;`

Result Grid | Filter Rows: | Edit: | Export/Import:

Patient_Id	Patient_Birthday	Patient_Gender	Doctor_Id
00a05408-8291-4231-886e-13763e103161	2021-03-12	F	45854
00a05408-8291-4231-886e-13763e103161	2021-03-13	F	45854
00c0b293-48e7-4e16-ac76-9269ba535a62	2021-03-10	F	45854
00d7c36e-3cdf-4df6-ac03-6c30cdc8e85b	2021-03-12	F	45854
00dd08bb-b7ea-4617-801e-02b3051c3475	2021-03-10	F	45854
00f3a73b-1801-43f7-992d-38338ec58a90	2021-03-11	M	56445

Enfin, l'application appelle le modèle à convolution et lance la prédiction, pour afficher le résultat de la classification, avec le pourcentage de probabilité sur la page “result”. Cette étape cette fois permet de peupler la table “radio”.

54 • `select * from radio;`

Result Grid | Filter Rows: | Export: | Wrap Cell Content: `IA`

Patient_Id	File_Path	Date	Etat	Probability
00dd08bb-b7ea-4617-801e-02b3051c3475	./uploads/00dd08bb-b7ea-4617-801e-02b3051...	2021-03-13	No Lung Opacity but Not Normal	83.67%
00d7c36e-3cdf-4df6-ac03-6c30cdc8e85b	./uploads/00d7c36e-3cdf-4df6-ac03-6c30cdc8e...	2021-03-20	No Lung Opacity but Not Normal	93.78%
00a05408-8291-4231-886e-13763e103161	./uploads/00a05408-8291-4231-886e-13763e1...	2021-03-04	No Lung Opacity but Not Normal	99.96%
00d7c36e-3cdf-4df6-ac03-6c30cdc8e85b	./uploads/00d7c36e-3cdf-4df6-ac03-6c30cdc8e...	2021-03-05	No Lung Opacity but Not Normal	93.78%
00dd08bb-b7ea-4617-801e-02b3051c3475	./uploads/00dd08bb-b7ea-4617-801e-02b3051...	2021-03-11	No Lung Opacity but Not Normal	83.67%
00dd08bb-b7ea-4617-801e-02b3051c3475	./uploads/00dd08bb-b7ea-4617-801e-02b3051...	2021-03-11	No Lung Opacity but Not Normal	83.67%

L'affichage de la totalité des résultats présents sur la table "radio" du "patient\_Id" en question sur cette page permet d'avoir un historique et donc un suivi plus précis d'éventuelles évolutions de la maladie pour le médecin.

De plus, l'application extrait l'image du DICOM et l'enregistre temporairement dans un dossier static, pour pouvoir l'afficher dans les résultats. Ainsi, cela permet un double contrôle, avec l'avis du médecin, basé sur la lecture de la radio.

Cela nous amène à une évolution possible : Nous pouvons envisager une fonction permettant au médecin d'approuver ou de contester la prédiction de l'IA, et ainsi affiner notre dataset avec des features supplémentaires, qui permettront de meilleures performances, dans un processus de ré-entraînement cyclique.

← → ↻ ⓘ 127.0.0.1:5000/result

NeumoniApp Prediction New prediction

64.11% Normal

Patient ID	Date du cliché	Diagnostic	Probabilité
00c0b293-48e7-4e16-ac76-9269ba535a62	2021-03-12	Normal	64.11%
00c0b293-48e7-4e16-ac76-9269ba535a62	2021-03-12	Normal	64.11%
00c0b293-48e7-4e16-ac76-9269ba535a62	2021-03-12	Normal	64.11%
00c0b293-48e7-			

A partir de cette page, il est ensuite possible de revenir à une nouvelle prédiction, ou de se déconnecter.

## Conclusion

Ce projet m'aura permis de mettre en pratique une grande partie des notions abordées durant cette formation Simplon. J'ai pu démarrer un projet qui me tenait à cœur, à savoir l'utilisation de l'IA dans le domaine médical, mais aussi de balayer toutes les étapes d'un projet IA, de la récupération des données, jusqu'à l'élaboration d'une interface client.

On aura vu ici l'importance et l'utilité du transfer learning, qui permet d'améliorer grandement les performances d'un modèle, tout en réduisant drastiquement la consommation de ressources de calcul.

Enfin, la création d'une base de données couplée à une interface utilisateur offre de nombreuses perspectives d'améliorations, notamment le stockage des modèles IA, le réentraînement planifié des modèles avec les nouvelles données labellisées par le praticien à la suite d'une prédiction.