Synthèse du Projet

ZOIDBERG 2.0

Détection de la Pneumonie

par apprentissage automatique

sur images de rayons X

Présenté par IA EPITECH

Thomas MAYOR Bastien FAUVEAU Paulin FOURQUET Adnane REKIBI Nikita OFMAN

Sommaire

Objectif du Projet

Le projet **ZOIDBERG 2.0** vise à développer un système d'aide au diagnostic de la pneumonie à partir d'images radiographiques thoraciques (rayons X), en utilisant des techniques d'apprentissage automatique. Trois jeux de données distincts sont exploités pour entraîner, valider et tester les modèles, avec l'objectif d'obtenir un outil fiable pour assister les médecins dans la détection de la pneumonie, et idéalement dans la distinction entre pneumonie bactérienne et virale.

I. Analyse initiale détaillée du dataset

1. Composition brute et déséquilibre des classes

- Le dataset initial contient **5858 images** réparties en deux classes principales :
    - **NORMAL** : 1583 images (≈27% du total)
    - **PNEUMONIA** : 4275 images (≈73% du total)

- Cette répartition montre un **déséquilibre important**, avec la classe PNEUMONIA environ **2,7 fois plus représentée** que la classe NORMAL.

- Ce déséquilibre est problématique car un modèle entraîné directement sur ces données risque de :
    - Favoriser la prédiction de la classe majoritaire (PNEUMONIA).
    - Négliger les caractéristiques de la classe minoritaire (NORMAL).
    - Obtenir une précision globale trompeusement élevée en raison de la surreprésentation de PNEUMONIA, mais avec une faible sensibilité pour NORMAL.

2. Analyse de la distribution des images dans les splits initiaux

- Avant équilibrage, la distribution des images dans les dossiers **train**, **validation** et **test** est elle aussi déséquilibrée, reflétant la composition globale.
- Par exemple, dans le dossier d'entraînement, on observe plus d'images PNEUMONIA que NORMAL, ce qui peut amplifier le biais lors de l'apprentissage.
- Cette répartition non équilibrée dans chaque split compromet la capacité à évaluer correctement la performance du modèle, notamment sur la classe minoritaire.

3. Analyse qualitative des images

- Les images NORMAL présentent une variabilité modérée en termes de luminosité, orientation, et qualité.
- Les images PNEUMONIA sont plus hétérogènes, incluant différentes formes de pneumonie (bactérienne, virale), avec des manifestations radiologiques variées.
- Certaines images contiennent des artefacts (marqueurs, tubes, annotations) qui peuvent influencer l'apprentissage si non traités.

4. Risques liés au déséquilibre et à la qualité des données

- Le déséquilibre peut entraîner un **biais de classification**, où le modèle prédit majoritairement la classe PNEUMONIA.
- Un modèle non équilibré peut avoir une **fausse impression de performance élevée** (accuracy élevée mais faible rappel sur NORMAL).
- La diversité et la qualité des images doivent être prises en compte pour éviter un sur-apprentissage sur des caractéristiques non pertinentes.

Dans notre analyse, on voit clairement la différence d'images entre les deux classes NORMAL et PNEUMONIA, nous avons plus de scanners de pneumonies que de scanners avec des poumons sains.

Il faut rééquilibrer les scanners sans pneumonie. Nous allons à présent passer à l'équilibrage et la justification de nos données pour pouvoir tester nos différents modèles.

Nous allons donc utiliser le même dataset commun afin de comparer lequel de nos modèles sera le plus précis et efficace dans notre solution d'analyse.

II.Préparation des données

Pour pallier ces problèmes, un **équilibrage a été nécessaire**.

Nous ne pouvons pas avoir de nouvelles images réelles de scanners, nous avons utilisé le principe de la **Data Augmentation**.

Cela consiste à appliquer des transformations simples à des images existantes pour en créer de nouvelles variantes réalistes.

Nous sommes donc passés sur la classe **NORMAL** de **1583** à environ **3500** images, en générant **1915 images** par transformations.

Voici les différents types d'augmentations utilisés :

- flip : Miroir horizontal (image retournée gauche ↔ droite)
- rotate10 : Rotation de l'image de +10°
- rotate-10 : Rotation de l'image de -10°
- zoom : On découpe une région centrale (90%) puis on la redimensionne à la taille originale (effet zoom)
- shift : On décale l'image vers la droite et le bas
- brightness : On augmente la luminosité (effet éclaircissement)

Une fois les images créées pour la classe **NORMAL**, nous avons préparé les **3500 images pour nos deux classes.**

Voici ce que nous récupérons :

- **NORMAL** : Nous avons rassemblé toutes les images de cette classe (1585 originales + 1915 augmentées = 3500).
- **PNEUMONIA** : Pour celle-ci, nous avons établi un sous-échantillonnage aléatoire de 3500 images pour correspondre à la classe **NORMAL** augmentée.

Cette démarche permet d'obtenir un dataset **équilibré** de 7000 images, avec une répartition équilibrée dans chaque split.

La répartition finale a été fixée à :

- **Train : 70%** (2450 images par classe)
- **Validation : 15%** (525 images par classe)
- **Test : 15%** (525 images par classe)

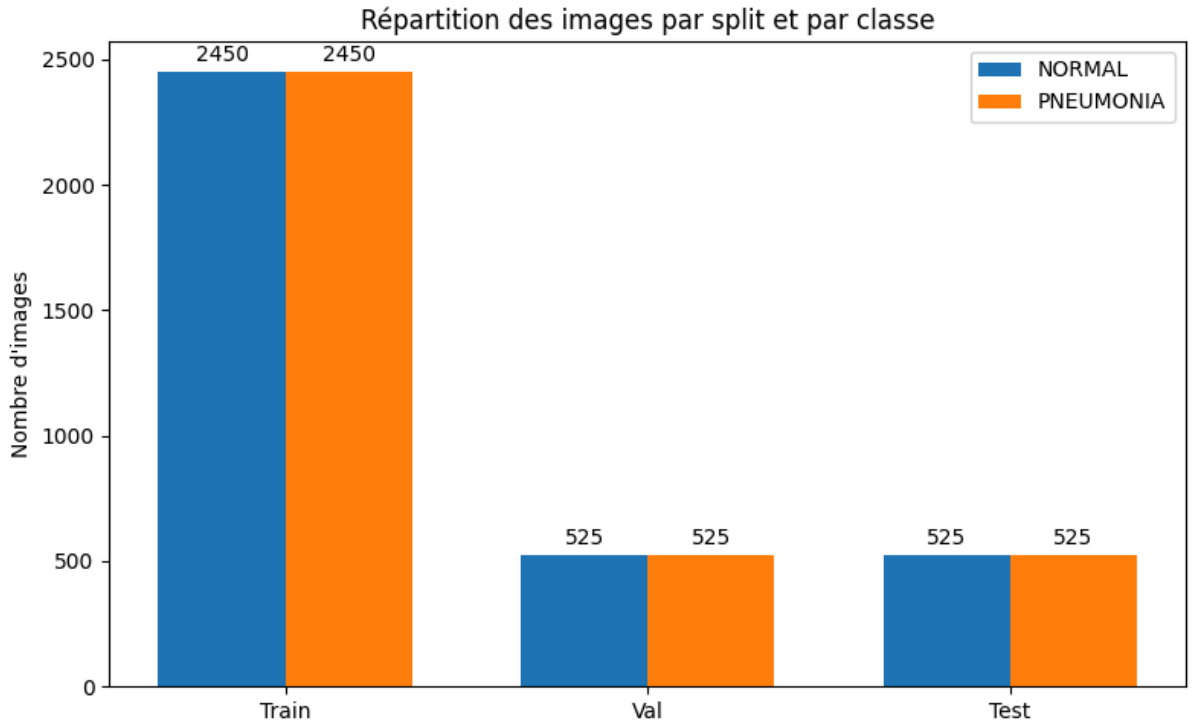
Split	Pourcentage	Normal	Pneumonia	Total
Train	70%	2 450	2 450	4 900
Val	15%	525	525	1 050
Test	15%	525	525	1 050

Cette répartition garantit que chaque phase d'entraînement et d'évaluation est réalisée sur des données représentatives et équilibrées, ce qui est essentiel pour une évaluation fiable et robuste des modèles.

Elle a été choisie dans ce raisonnement :

- Il faut donner beaucoup de données au modèle pour apprendre donc on vise les **70%** du dataset **pour l'entraînement**
- Il faut aussi garder assez de données pour valider et tester donc pour éviter le surapprentissage on prend **15% du dataset pour la validation**
- et **les 15% restants vont au dataset pour les tests** pour évaluer objectivement les performances de notre modèle.

C'est particulièrement utile avec **un jeu de données de taille moyenne comme sur cette analyse.** Cela garantit un modèle **fiable, robuste, et équilibré**, surtout important dans un contexte médical sensible.



Il faut à présent passer à l'étape suivante qui n'est qu'autre que le **Prétraitement des images** autrement appellé "**Preprocessing Data"** mais nous allons dans un premier temps expliqués les différents modèles utilisés pour cette analyse car nous n'avons pas pu avoir les mêmes paramètres selon nos modèles choisis.

III.Nos modèles

Chaque membre du groupe (5 au total) a choisi et entraîné un modèle différent pour comparer les performances :

Modèle	Type	Particularités techniques	Choix motivé
VGG16	CNN pré-entraîné	- Architecture simple et profonde - Pré-entraîné sur ImageNet - Fine-tuning - Couches denses en sortie	-Très utilisé en imagerie médicale pour sa **robustesse** et sa **simplicité**. - Fonctionne bien même sur de petits datasets.
Xception	CNN pré-entraîné	- Architecture **depthwise separable convolutions** (convolutions plus légères et plus rapides, tout en gardant une bonne performance, On utilise un seul filtre par canal de l'image (RGB), ce qui signifie que chaque canal est traité séparément et ne sont pas combinés.) - Très profonde - Fine-tuning	- Excellente performance sur les tâches complexes. - Utilise moins de paramètres que Inception, tout en étant **plus efficace**. - Très bon pour détecter de **petits détails médicaux**.

Random Forest	Modèle de machine learning classique (non CNN)	- Algorithme basé sur des **arbres de décision** - Fonctionne bien sur des **données tabulaires**	- Bon modèle de **référence de base (baseline)**. - Interprétable, rapide à entraîner.
DenseNet	CNN pré-entraîné (ImageNet)	- Connexions denses entre couches (chaque couche reçoit les sorties de toutes les précédentes) - Couches gelées au début	- Réduit la perte d'information, **favorise la propagation des gradients**. - Moins de paramètres, **meilleure efficacité**.
CNN from scratch	CNN construit manuellement	- Architecture personnalisée- Entièrement entraîné sur les données du projet- Fine-tuning possible	- Adapté aux données spécifiques si tu as un **dataset bien annoté et assez grand**. - Permet de **contrôler chaque détail du modèle**.

Ces modèles ont été choisis pour **diversifier les approches** :

- **CNN pré-entraînés (VGG16, Xception, DenseNet)** : profitent de **l'apprentissage déjà acquis sur ImageNet**, ce qui est utile quand les données médicales sont limitées.

- **Random Forest** : sert de **modèle comparatif classique** ; simple à interpréter, et permet de tester si un **modèle classique sans CNN** pourrait suffire avec de bons descripteurs.

- **CNN from scratch** : permet d'**adapter l'architecture à notre dataset spécifique**, et d'avoir **une maîtrise totale** du réseau.

IV.Prétraitement des données

L'objectif est d'homogénéiser, normaliser et améliorer la qualité des données pour de meilleures performances en deep learning.

1.Prétraitement spécifique à Xception

Lorsqu'on travaille avec un modèle comme **Xception**, il est essentiel d'appliquer un **prétraitement adapté**, pour garantir que les images soient au bon **format**, **type** et **échelle de valeurs**.

A.Redimensionnement à 299×299 pixels

Le modèle Xception attend des **images d'entrée** de **taille 299×299 pixels**.

Peu importe la taille d'origine des images (par exemple 1024×1024 ou 256×256), elles sont **redimensionnées** pour correspondre à l'entrée du réseau.
Cela permet :

- D'avoir un format d'entrée **standardisé**
- De **réduire la charge de calcul** sans trop perdre d'informations visuelles importantes

B.Conversion en RGB

Certaines images peuvent être en **niveau de gris** (1 canal), mais les modèles pré-entraînés comme Xception s'attendent à des images **RGB à 3 canaux**.
La conversion en RGB est **automatique** avec la fonction **image_dataset_from_directory** de TensorFlow/Keras.

Cela garantit que chaque image a bien **3 canaux de couleur (R, G, B)**, même si l'image source est en noir et blanc (ex : radiographie).

C.Normalisation spécifique à Xception : pixels entre -1 et 1

Les images brutes ont des valeurs de pixels entre **0 et 255**. Pour le modèle Xception, il faut appliquer la **fonction preprocess_input** fournie par Keras.

Cette fonction convertit les pixels dans l'intervalle **[-1, 1]**, ce qui est important pour :

- Respecter les conditions d'entraînement initiales du modèle
- **Accélérer la convergence** pendant l'entraînement
- Améliorer la **stabilité numérique** du réseau

D.Optimisation du pipeline

.prefetch() est une fonction de TensorFlow qui **accélère le traitement des données**. Elle permet au CPU de **préparer les images suivantes pendant que le GPU entraîne le modèle sur les images actuelles**.
Cela permet donc :

- Une réduction du **temps d'attente** entre les lots (batches)
- Une utilisation **efficace du matériel** (GPU et CPU en parallèle)

Il est **fortement recommandé** de visualiser quelques images après le prétraitement pour :

- Vérifier que les images sont bien en 299×299 pixels
- Vérifier qu'elles sont en couleur (3 canaux)
- S'assurer que la normalisation ne les a pas trop altérées visuellement

- Cela permet d'éviter des erreurs silencieuses et de **valider le bon fonctionnement du pipeline**.

2.Prétraitement spécifique au VGG16

A.Redimensionnement à 150×150 pixels

VGG16 a été conçu pour recevoir des images de taille 224×224 pixels en entrée. Cependant, il est courant d'adapter cette taille à 150×150 pour des raisons de :

- Capacité mémoire : Les images médicales (X-ray) sont souvent volumineuses. Travailler avec des images plus petites permet d'entraîner le modèle plus rapidement et d'utiliser des batchs plus grands, ce qui accélère la convergence.
- Performance : Sur des datasets de taille moyenne, une résolution de 150×150 conserve suffisamment d'informations discriminantes pour la détection de la pneumonie, tout en limitant le bruit et le surapprentissage.

- Compatibilité : Les couches convolutives de VGG16 peuvent accepter différentes tailles d'entrée si on retire la tête de classification (ce que vous faites dans votre pipeline). Cela permet une certaine flexibilité.
- Le modèle conserve la structure des patterns appris (bords, textures, formes) même sur des images redimensionnées, car les premières couches de convolution détectent des motifs locaux.
- Le choix de 150×150 est un compromis entre rapidité d'entraînement et préservation des détails médicaux importants.

B.Normalisation des pixels (rescale à 1/255)

- Les images X-ray sont en niveaux de gris, codées sur 8 bits (valeurs de 0 à 255).
- VGG16 (et la plupart des CNN pré-entraînés) ont été initialement entraînés sur ImageNet avec des images normalisées.
- Normaliser les pixels (diviser par 255) ramène les valeurs dans l'intervalle [0,, ce qui :
    - ➢ Facilite l'apprentissage (les gradients sont plus stables, la descente de gradient converge plus vite).
    - ➢ Évite les problèmes numériques liés à de grandes valeurs d'entrée.
    - ➢ Rend les données compatibles avec les poids pré-entraînés du modèle.

- Les poids des premières couches de VGG16 s'attendent à des valeurs d'entrée normalisées. Sans cette étape, le modèle aurait des activations incohérentes et les performances chuteraient drastiquement.

3.Prétraitement spécifique au CNN basique

Lorsque l'on travaille avec un modèle CNN construit manuellement (et non pré-entraîné), on a plus de liberté dans la définition du prétraitement. Voici les étapes détaillées appliquées à ton pipeline :

A.Redimensionnement des images

Toutes les images, quelle que soit leur taille d'origine, sont redimensionnées en **224×224 pixels** via **image_dataset_from_directory.**

Cela permet :

- D'uniformiser les dimensions des images pour l'entrée du modèle.

- De faciliter le traitement par les couches de convolution (qui attendent des tailles fixes).
- De réduire le temps d'entraînement tout en conservant une bonne résolution pour les détails radiographiques.

B.Conversion en niveaux de gris

Contrairement aux modèles pré-entraînés, ici nous travaillons avec des **images en niveaux de gris** (color_mode='grayscale'), car les radiographies pulmonaires sont monochromes.

Cela a plusieurs avantages :

- Réduction du **nombre de paramètres** (1 canal au lieu de 3).
- Adapté à la **nature des données médicales** (où la couleur est inutile).
- Amélioration potentielle de la **généralisation**, car le modèle se concentre uniquement sur les contrastes et structures internes.

C.Normalisation

Les images sont **normalisées** avec un **Rescaling(1./255)**, ce qui transforme chaque pixel (initialement entre 0 et 255) en une valeur **entre 0 et 1**.

Cette étape est cruciale pour :

- Faciliter la convergence du modèle.
- Éviter les instabilités numériques dues à des valeurs de pixels trop grandes.
- Avoir des gradients plus stables pendant l'entraînement.

A. Optimisation du pipeline

Pour optimiser les performances de chargement des données, plusieurs techniques sont combinées :

- **.cache()** : stocke les données en mémoire après la première lecture.
- **.shuffle(1000)** : mélange les images à chaque époque pour éviter le surapprentissage de l'ordre.
- **.prefetch(buffer_size=tf.data.AUTOTUNE)** : prépare les lots suivants pendant que le modèle s'entraîne sur le lot courant.

Avantages :

- Amélioration du débit GPU.
- Réduction du temps d'entraînement global.
- Meilleure efficacité CPU/GPU en parallèle.

4.Prétraitement spécifique au Random Forest (+ HOG/LBP)

Contrairement aux modèles de deep learning (CNN), Random Forest nécessite une approche différente car il ne travaille pas directement sur les pixels bruts des images. Il faut d'abord extraire des caractéristiques (features) des images pour les transformer en données tabulaires.

A. Redimensionnement uniforme

Étape obligatoire :

```
img = img.resize((224, 224))  # Toutes les images sont redimensionnées
```

Pourquoi ?

- Les images du dataset ont des tailles variables
- Random Forest nécessite que toutes les features aient la même dimension
- La taille 224×224 est un standard qui conserve suffisamment de détails pour l'extraction de caractéristiques

B. Vérification du format des images

Traitement conditionnel : Les fonctions d'extraction vérifient automatiquement si l'image a 3 canaux (RGB) ou 1 canal (niveaux de gris). Si l'image est en couleur, elle est convertie en niveaux de gris pour l'extraction des caractéristiques.

Pourquoi cette vérification ?

- Les radiographies peuvent être sauvegardées en RGB (3 canaux) même si elles sont visuellement en niveaux de gris
- Les caractéristiques HOG et LBP nécessitent des images monochromes
- Assure la compatibilité avec différents formats d'images du dataset

C. Extraction de caractéristiques HOG (Histogram of Oriented Gradients)

Paramètres configurés :

- 9 orientations de gradients pour capturer différentes directions
- Cellules de 16×16 pixels pour analyser les détails locaux
- Blocs de 2×2 cellules pour la normalisation locale
- Normalisation L2-Hys pour la robustesse

- Transform_sqrt activé pour améliorer la stabilité

Objectif :

- Capture les contours et formes dans l'image
- Détecte les patterns de pneumonie (opacités, consolidations)
- Robuste aux variations d'illumination

D. Extraction de caractéristiques LBP (Local Binary Pattern)

Configuration :

- P=24 points échantillonnés sur un rayon R=3
- Méthode 'uniform' pour réduire la dimensionnalité
- Création d'un histogramme normalisé des patterns locaux

Objectif :

- Analyse la texture locale de l'image
- Détecte les variations de texture caractéristiques de la pneumonie
- Complément aux caractéristiques HOG pour une analyse plus fine

E. Combinaison des caractéristiques (HOG + LBP)

Fusion des descripteurs : Les caractéristiques HOG et LBP sont concaténées pour créer un vecteur de features plus complet combinant l'information de forme et de texture.

Avantages :

- Représentation plus riche de l'image
- Complémentarité entre forme (HOG) et texture (LBP)
- Améliore les performances de classification

F. Pas de normalisation pixel classique

Différence avec les CNN :

- Pas de division par 255 : Les algorithmes HOG/LBP gèrent directement les valeurs 0-255
- Pas de normalisation [-1,1] : Non nécessaire car on extrait des features, pas des pixels
- Normalisation intégrée : Les algorithmes HOG et LBP incluent leurs propres normalisations

G. Dimensions finales des caractéristiques

Après prétraitement, chaque image est représentée par :

- HOG seul : ~1764 caractéristiques
- LBP seul : 26 caractéristiques (histogramme)
- HOG + LBP : ~1790 caractéristiques

Avantages de cette approche

1. Interprétabilité : Les caractéristiques HOG/LBP sont explicables médicalement
2. Robustesse : Moins sensible au surapprentissage que les CNN
3. Rapidité : Entraînement très rapide comparé au deep learning
4. Efficacité mémoire : Pas besoin de GPU, fonctionne sur CPU standard

Limites

1. Perte d'information : Réduction drastique de l'information (224×224×3 = 150,528 pixels → ~1790 features)
2. Features manuelles : Contrairement aux CNN qui apprennent automatiquement les features optimales
3. Moins de nuances : Peut manquer des patterns complexes détectables par le deep learning

5.Prétraitement spécifique au model DenseNet 121

A.Redimensionnement des images à 224x224 pixels

Le modèle **DenseNet121**, utilisé ici avec des poids pré-entraînés sur ImageNet, attend des images de taille 224×224 pixels. Il est donc essentiel de redimensionner les radiographies pulmonaires à cette taille pour garantir la compatibilité avec la structure du réseau.

Le redimensionnement présente plusieurs avantages :

- Il assure la compatibilité directe avec DenseNet121, sans modification de l’architecture.

- Il permet une meilleure efficacité mémoire et un entraînement plus rapide.

- Il conserve suffisamment de détails médicaux pour la tâche de classification (présence ou non de pneumonie).

DenseNet121, grâce à ses couches convolutives profondes et denses, est capable d’extraire des motifs pertinents même à partir d’images redimensionnées. Cela permet de détecter des éléments importants tels que des opacités ou anomalies dans les poumons.

B.Normalisation des pixels

Les images médicales utilisées sont en niveaux de gris, codés sur 8 bits, avec des valeurs comprises entre 0 et 255. Pour être compatibles avec DenseNet121 pré-entraîné sur ImageNet, les images doivent être normalisées.

Cette normalisation est effectuée en divisant chaque pixel par 255, ce qui ramène les valeurs dans l'intervalle [0, 1].

Cette étape est indispensable pour plusieurs raisons :

- Elle aligne les données d'entrée avec les données sur lesquelles le modèle a été initialement entraîné.
- Elle stabilise les gradients et facilite l'apprentissage.
- Elle permet une convergence plus rapide et évite les problèmes numériques liés à de grandes valeurs en entrée.

C.Structuration des données

Les données ont été organisées en trois sous-dossiers : entraînement, validation et test. La fonction **image_dataset_from_directory** de TensorFlow a été utilisée pour :

- Charger automatiquement les images avec leurs étiquettes respectives (NORMAL ou PNEUMONIA).
- Créer des ensembles de données optimisés avec des options comme le batching, le préchargement et l'optimisation automatique (AUTOTUNE).
- Appliquer le redimensionnement et la normalisation directement dans le pipeline de traitement.

Ce pipeline assure un flux de données cohérent, efficace et adapté à l'entraînement d'un modèle profond sur un jeu de données médical.

V.Entraînement des données

Dans ce projet, plusieurs modèles sont entraînés pour comparer leurs performances sur la même tâche. Afin d'assurer une **comparaison équitable**, tous les modèles partagent certaines **conditions d'entraînement communes**.

1.Conditions d'entraînement pour les réseaux de neurones

- **Le nombre d'époques** : Tous les modèles basés sur des réseaux de neurones sont entraînés pendant **10 époques** hormis **Random Forest** qui n'est pas un réseau de neurones.

- **Batch size :** Dans ce projet, nous avons utilisé un **batch size de 32**.
  Cela signifie que **32 images** sont traitées **à la fois** avant que le modèle ne mette à jour ses poids.

Pourquoi avoir choisi 32 ?

➔ **Équilibre entre performance et mémoire** :

- ◆ Un batch size trop petit (ex : 8) rend l'apprentissage plus **bruyant** et **instable**.
- ◆ Un batch size trop grand (ex : 128) est **plus stable**, mais demande **plus de mémoire GPU** et **peut ralentir la convergence**.

➔ **32** est une **valeur standard** souvent utilisée dans la pratique : elle est **assez efficace** sur la plupart des architectures (CNN, VGG, Xception…).

➔ Elle permet un **temps d'entraînement raisonnable** sans sacrifier la performance.

- **Nombre d'itérations par époque :** Le **nombre d'itérations** dépend de la taille du dataset et du batch size. Ce nombre correspond au **nombre de fois que le modèle met à jour ses poids pendant une époque**.

  Le nombre d'itérations influence **la fréquence de mise à jour du modèle**. Il permet aussi de **comparer le temps d'entraînement** entre plusieurs modèles.
  Dans les logs d'entraînement (fit()), chaque époque affiche ces **50 itérations**, chacune avec son loss et accuracy.

Le dataset contient 7000 images réparties en 70% pour l'entraînement (4900), 15% pour la validation (1050) et 15% pour le test (1050).

Un batch size de 32 a été utilisé, ce qui donne environ **153 itérations par époque** pour l'entraînement, et **33 pour la validation et le test**.

Ce choix permet un bon **compromis entre stabilité d'apprentissage, efficacité et utilisation mémoire**, tout en assurant une couverture suffisante du dataset à chaque époque.

2.Fine tuning

Le fine-tuning, ou ajustement fin, est une technique utilisée en deep learning pour améliorer un modèle pré-entraîné (souvent sur un grand dataset comme ImageNet) en l'adaptant à un nouveau jeu de données spécifique.

On commence généralement par « geler » les premières couches du modèle (qui détectent des motifs généraux comme les bords et les textures), puis on « déverrouille » progressivement les couches profondes, plus spécialisées, afin de les réentraîner sur notre propre jeu de données.

Cela permet de **réutiliser les connaissances déjà apprises** tout en **affinant** les couches les plus pertinentes pour la tâche cible, avec un **learning rate plus faible** pour éviter de déstabiliser les poids existants.

A.Xception

1. **Déblocage progressif des couches**

Nous avons rendu **seulement les 12 dernières couches du modèle Xception entraînables**, car elles sont responsables des **caractéristiques de haut niveau**, plus spécifiques à notre jeu de données.

Le reste du modèle reste gelé pour **conserver les connaissances générales pré-apprises** sur ImageNet.

2. **Recompilation du modèle**

    - Le modèle est recompilé avec un **learning rate très faible** (2e-5) pour **éviter de détruire les poids pré-entraînés**. On utilise l'Optimiseur Adam qui ajuste automatiquement la vitesse à laquelle les poids du modèle sont mis à jour pendant l'entraînement.

    - La fonction de perte (binary_crossentropy) et la métrique (accuracy, precision) sont conservées.

3. **Poursuite de l'entraînement**

    - L'entraînement continue à partir de l'époque 5, mais avec **moins d'époques**, car le fine-tuning demande **des ajustements plus subtils** que l'entraînement initial.

    - Nous utilisons des **callbacks** pour un contrôle optimal :

- **EarlyStopping** : évite le surapprentissage.
- **ReduceLROnPlateau** : ajuste dynamiquement le learning rate.
- **ModelCheckpoint** : enregistre les meilleurs poids du modèle.

Ensuite, un **fine-tuning progressif** est réalisé :

- Les **12 dernières couches** du modèle de base sont **débloquées**.
- Cela permet d'ajuster les couches les plus spécialisées.

B.DenseNet121

Étapes :

- Le modèle est initialement **gelé** pour entraîner uniquement la nouvelle tête.
- Puis, **50 dernières couches** du backbone sont **débloquées**.

- **Tête de classification** :
  - GlobalAveragePooling2D
  - Dense(128, relu) + Dropout
  - Dense(1, sigmoid)
- **Compilation** :
  - Optimiseur Adam avec learning rate **faible** : 1e-5
  - loss = 'binary_crossentropy'
  - Métrique : accuracy
- **Callbacks utilisés** :
  - EarlyStopping
  - ModelCheckpoint

C.VGG16

On importe VGG16 sans la tête (include_top=False). Nous avons débloqué les **4 dernières couches** convolutives. Tout le reste est **gelé.**

- **Tête de classification** :
    - Flatten → Dense(256, relu) → Dropout(0.5) → Dense(1, sigmoid)
- **Compilation** :
    - Optimiseur Adam lr = 1e-4
    - Perte : binary_crossentropy
    - Métrique : accuracy
- **Pas de callbacks explicites mentionnés**, mais fortement recommandés

D.CNN

- **Pas de fine-tuning**, car le modèle est entraîné **from scratch**
- **Architecture classique** :
    - 3 blocs Conv2D + MaxPooling
    - Flatten → Dense(64) → Dropout → Dense(1, sigmoid)
- **Compilation** :
    - Optimiseur : Adam
    - Perte : binary_crossentropy
    - Métriques :
        - accuracy
        - AUC, Precision, Recall
- **Callbacks utilisés** :

    - EarlyStopping
    - ReduceLROnPlateau

E.Random Forest

- Pas de fine-tuning de réseau de neurones, mais optimisation d'hyperparamètres
- **Technique** :
    - Utilisation de GridSearchCV avec validation croisée cv=5
    - Recherche des meilleurs hyperparamètres :
        - n_estimators, max_depth, min_samples_split, min_samples_leaf
    - Scoring basé sur accuracy
- **Durée mesurée**, modèle le plus performant sauvegardé (pickle)

3.Métriques choisies pour l'évaluation des modèles

Dans le cadre de ce projet, nous avons retenu deux métriques principales pour évaluer les performances de nos modèles.

A.Accuracy (précision globale)

Il s'agit du **taux de bonnes prédictions** parmi l'ensemble des prédictions faites par le modèle. Elle indique **à quel point le modèle classe correctement** les images.

B.Loss (fonction de perte)

C'est une mesure de **l'erreur** entre les prédictions du modèle et les vraies étiquettes.

Dans notre cas, on utilise la **binary_crossentropy** car il s'agit d'un **problème de classification binaire** (deux classes). Elle sert à **guider l'apprentissage** : le modèle ajuste ses poids pour **minimiser cette erreur**.

Plus la loss est **basse**, meilleur est le modèle.

C.Pourquoi le choix de ces deux métriques ?

La loss donne une information fine sur la progression de l'apprentissage et la capacité du modèle à ajuster ses prédictions.

La précision donne une lecture directe de la performance sur la tâche de classification, facilement compréhensible et exploitable.

Les deux sont complémentaires : la loss peut continuer à baisser même si la précision stagne, ce qui peut signaler un surapprentissage ou une optimisation trop fine sur les données d'entraînement.

On choisit la loss pour piloter l'apprentissage du modèle, et la précision pour évaluer sa performance concrète sur la tâche de classification.
Les deux courbes sont nécessaires pour bien diagnostiquer le comportement du modèle à chaque étape de l'entraînement.

4.Métriques choisies pour le jeu de test

A.La matrice de confusion

Elle permet de visualiser la performance du modèle en détail. Contrairement à une simple mesure d'exactitude (accuracy), elle offre une vue granulaire sur :

Les vrais positifs (TP) : ici, les cas de pneumonie bien prédits.

Les vrais négatifs (TN) : les cas normaux bien identifiés.

Les faux positifs (FP) : les cas normaux classés à tort comme pneumonie.

Les faux négatifs (FN) : les cas de pneumonie non détectés.

Cela permet de mieux évaluer les erreurs du modèle, ce qui est crucial dans des cas sensibles comme la détection de maladies (ex: éviter les faux négatifs en santé).

B.La courbe ROC (Receiver Operating Characteristic)

Elle est utilisée pour évaluer les performances d'un modèle de classification binaire à différents seuils de décision. Elle trace le taux de vrais positifs (sensibilité) contre le taux de faux positifs. Son intérêt :

Elle permet de comparer plusieurs modèles indépendamment du seuil.

L'aire sous la courbe (AUC) est un indicateur global de performance :

AUC ≈ 0.5 : modèle aléatoire.

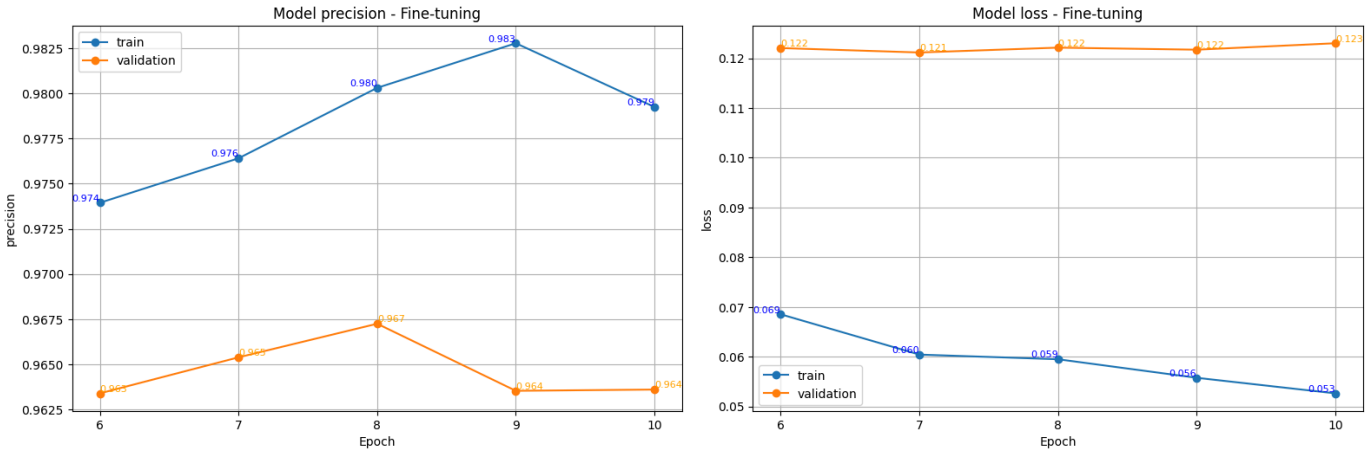
AUC = 1.0 : modèle parfait.

AUC ≥ 0.9 : modèle excellent.

VI.Comparaisons

A.Interprétation des résultats du modèle Xception

1.Courbes d'entraînement (Précision/Loss)



- **Courbe de précision** :

➔ **Train** : Augmente globalement de 0.974 à 0.983 entre l'époque 6 et 9, puis légère baisse à 0.979 à l'époque 10.
➔ **Validation** : Relativement stable, fluctue entre 0.963 et 0.967.

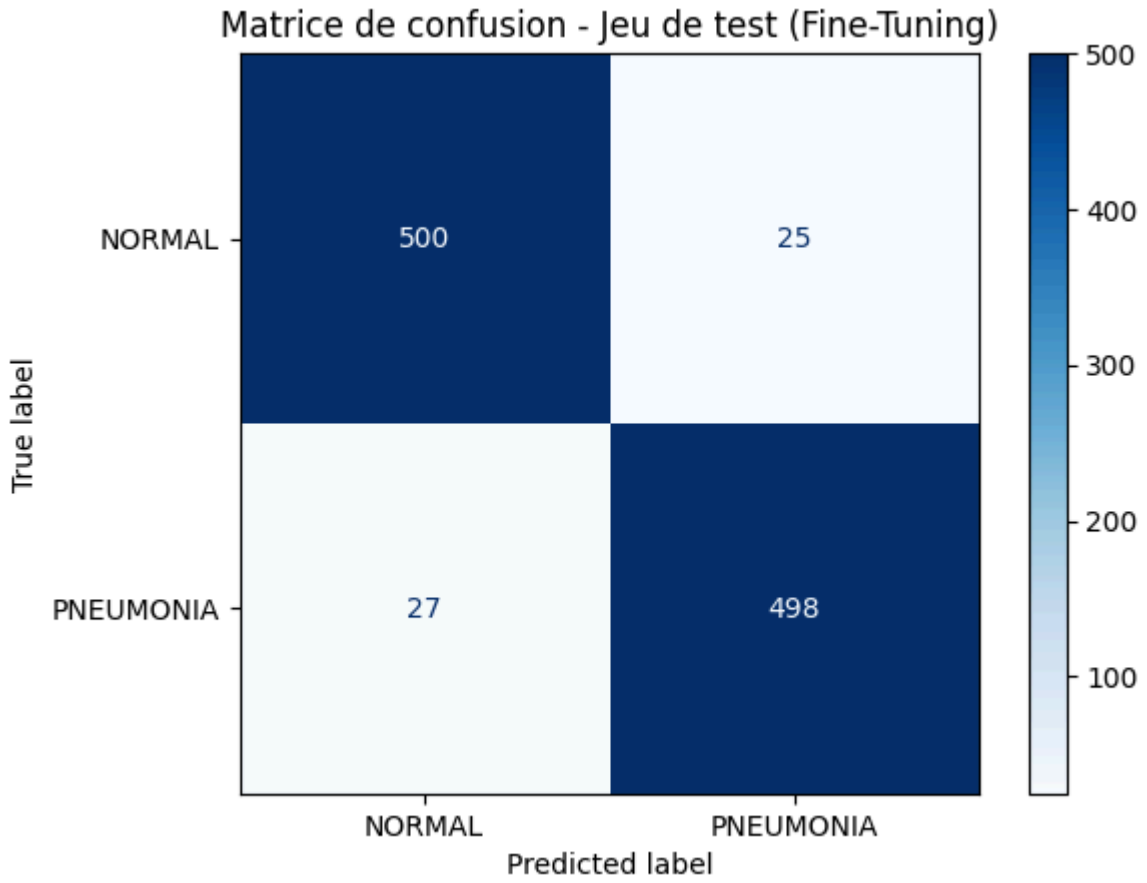
Le modèle continue d'apprendre sur les données d'entraînement (hausse de la précision). Mais la précision de validation plafonne, ce qui indique un début de surapprentissage (overfitting) dès l'époque 9-10.

- Courbe Loss :

➔ **Train :** Diminue régulièrement (de 0.069 à 0.053), ce qui montre que le modèle s'ajuste bien.
➔ **Validation :** Stable autour de 0.121–0.123

Le gap entre perte train/validation indique que le modèle surapprends sur les données d'entraînement à partir de l'époque 9. **Le meilleur compromis se situe autour de l'époque 8.**

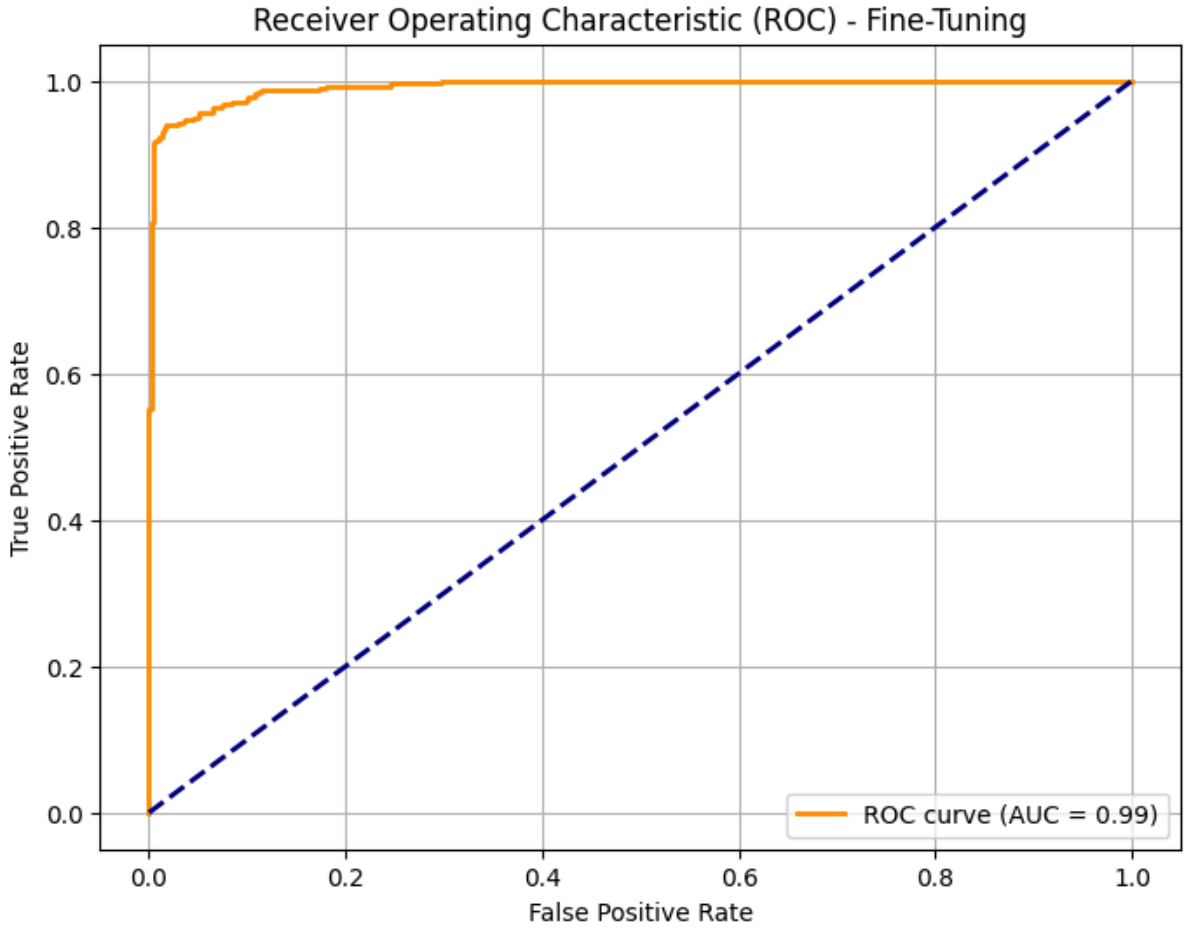
2.Matrice de confusion



Le modèle confond encore 25 NORMAL et 27 PNEUMONIA, mais c'est peu par rapport à la taille totale.

La performance est très équilibrée sur les deux classes → pas de biais marqué.

3.Courbe ROC



L'AUC = 0.99 indique une excellente capacité discriminante du modèle.

Cela signifie que dans 99% des cas, le modèle classe correctement une image aléatoire de chaque classe.

Le modèle montre une excellente précision globale (presque 97.4% d'exactitude brute) avec un très bon équilibre entre sensibilité et spécificité.

Une performance robuste avec un AUC proche de 1.

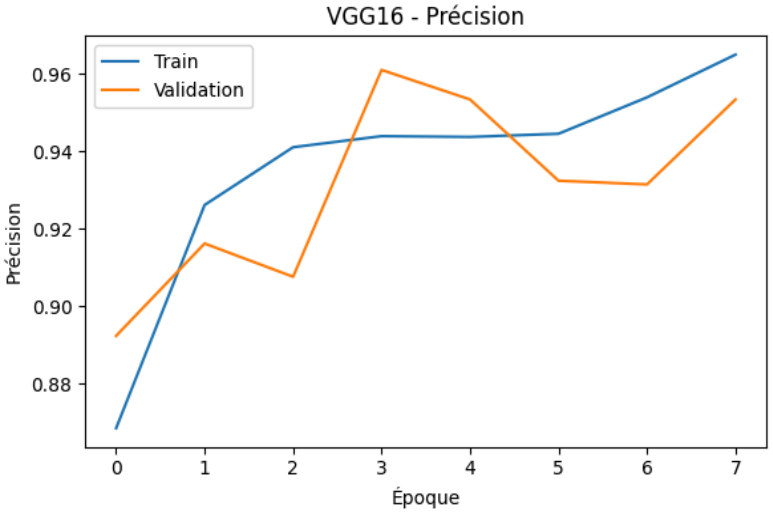
Cela confirme que le modèle est très performant pour la classification binaire entre les cas de pneumonie et les cas normaux.

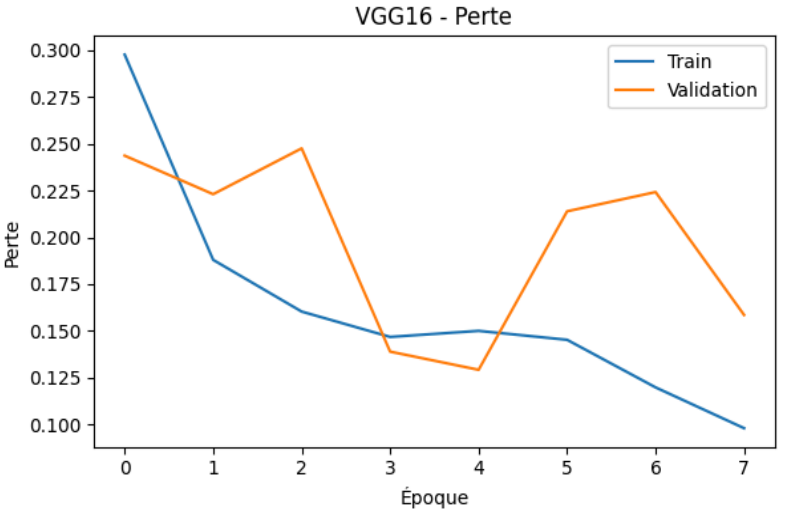
Même si la précision sur validation est stable, le modèle reste robuste en termes de classification binaire.

Cela suggère que le modèle est fiable pour une mise en production, avec éventuellement une régularisation à considérer pour éviter l'overfitting.

B.Interprétation des résultats du modèle VGG16

1.Courbes d'entraînement (Précision/Loss)





Courbe de précision :

- **Train** : Progression rapide de 0.87 à 0.97 sur 7 époques, avec une croissance quasi-linéaire constante
- **Validation** : Évolution erratique entre 0.89 et 0.96, avec des fluctuations importantes (pic à l'époque 3, chute à l'époque 2, remontée finale)

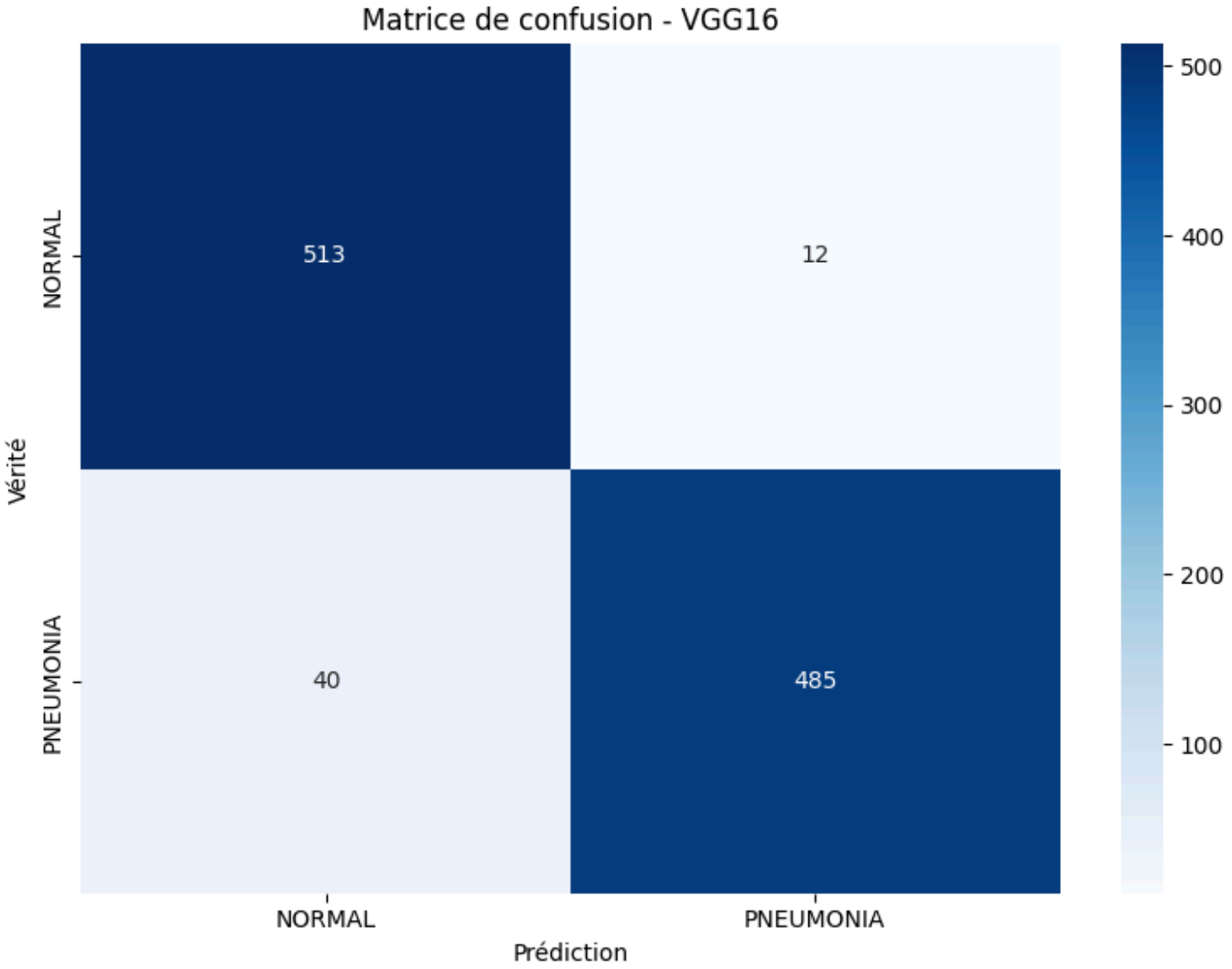
Analyse : Le modèle apprend efficacement sur les données d'entraînement avec une progression stable. Cependant, les fluctuations importantes de la validation suggèrent une instabilité du modèle et un début de surapprentissage dès l'époque 4-5. Le meilleur compromis semble se situer vers l'époque 3.

Courbe Loss :

- **Train** : Diminution régulière et constante de 0.30 à 0.10, montrant un apprentissage optimal
- **Validation** : Fluctuations marquées entre 0.13 et 0.25, avec une tendance à la hausse après l'époque 3

Analyse : L'écart croissant entre les pertes train/validation après l'époque 3 confirme le surapprentissage. Le modèle mémorise les données d'entraînement au détriment de la généralisation.

2.Matrice de confusion

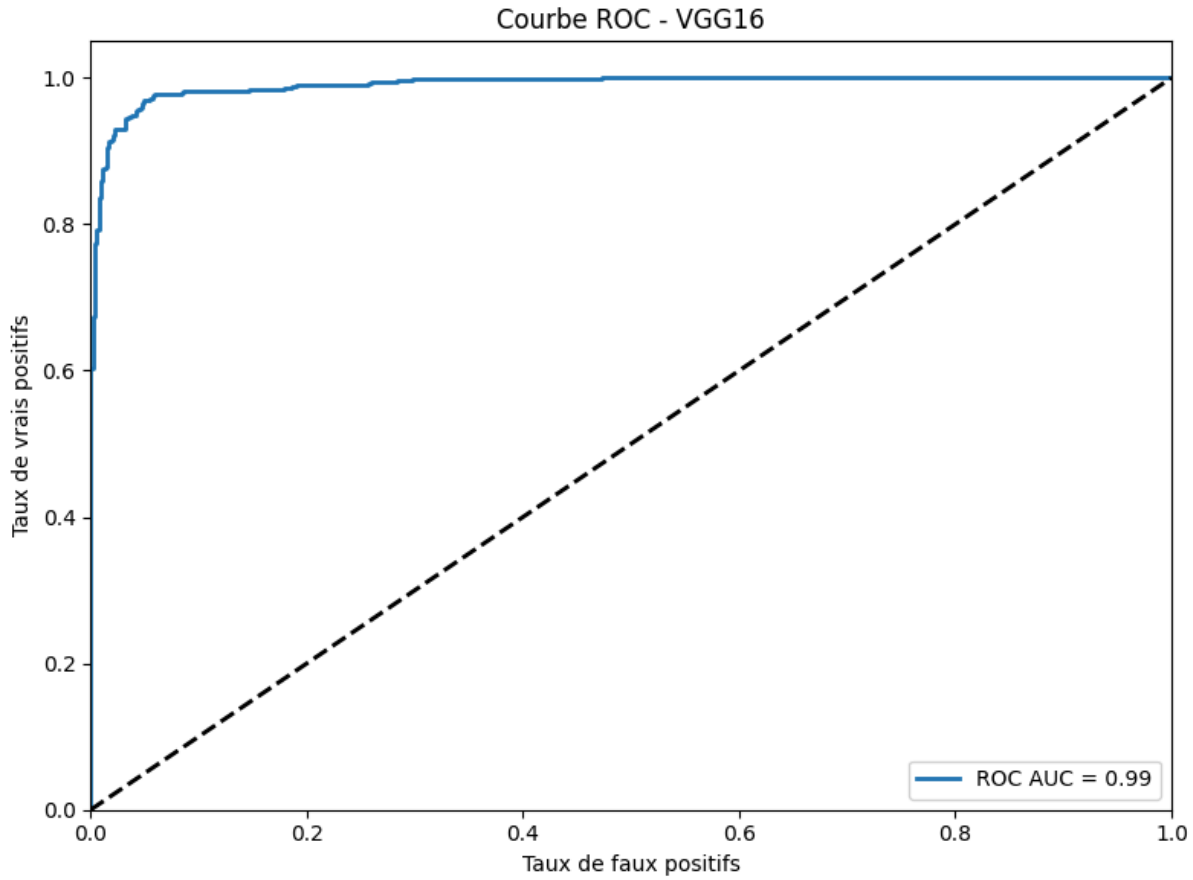


Performance équilibrée :

- Vrais positifs : 513 NORMAL correctement classifiés
- Vrais négatifs : 485 PNEUMONIA correctement classifiés
- Faux positifs : 12 NORMAL classifiés comme PNEUMONIA
- Faux négatifs : 40 PNEUMONIA classifiés comme NORMAL

Analyse : Excellente performance avec seulement 52 erreurs sur 1050 images (≈95% d'exactitude). Le modèle montre un léger biais vers la classe NORMAL (plus de faux négatifs que de faux positifs), ce qui pourrait être critique en diagnostic médical.

3.Courbe ROC



AUC = 0.99 indique une capacité discriminante exceptionnelle :

- Dans 99% des cas, le modèle classe correctement une paire d'images aléatoires
- La courbe suit parfaitement le coin supérieur gauche, signe d'un modèle quasi-optimal

Synthèse générale

Points forts :

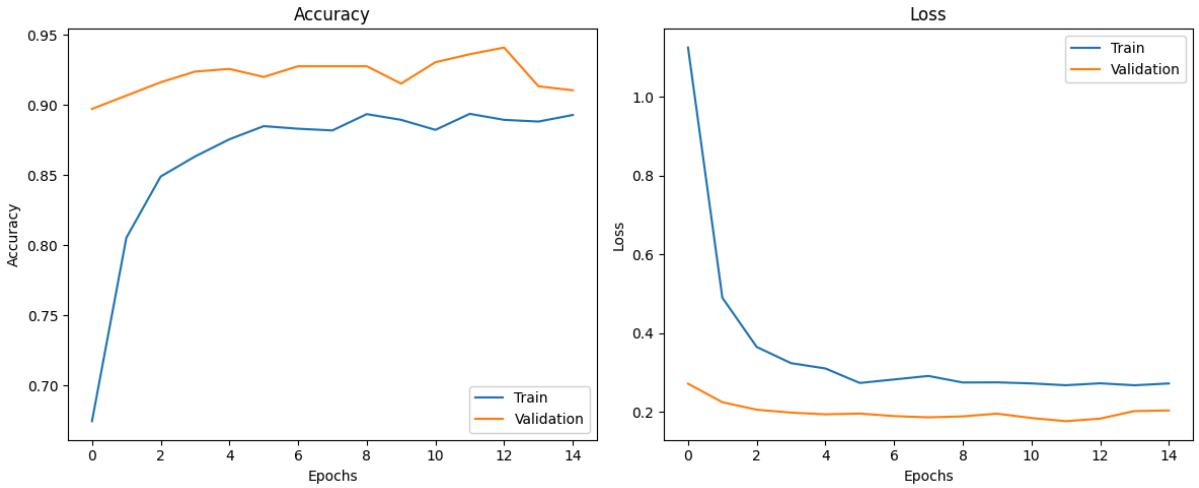
- Performance globale excellente (≈95% d'exactitude)
- AUC exceptionnelle (0.99) confirmant la robustesse discriminante
- Équilibre relatif entre les classes malgré un léger biais

Points d'attention :

- Signes clairs de surapprentissage à partir de l'époque 4
- Instabilité sur les données de validation (fluctuations importantes)
- Léger biais diagnostique (plus de faux négatifs) potentiellement problématique en contexte médical

C.Interprétation des résultats du modèle DenseNet

1.Courbes d'entraînement (Précision/Loss)



Graphique sur 15 Epoch avant Fine-Tuning

- **Courbe de précision** :

➔ L'accuracy de validation atteint environ 0.94 après 15 epochs.

➔ On observe une stabilisation voire une légère baisse après l'epoch 12, ce qui peut indiquer un début de sur-apprentissage

➔ L'accuracy d'entraînement monte progressivement (~0.89), mais reste toujours inférieure à celle de validation signe d'un modèle régularisé mais potentiellement sous-optimal.

- Courbe Loss :

➔ La courbe de perte diminue au fil des epochs, mais tend à se stabiliser autour de 0.2 pour la validation.

➔ La perte d'entraînement reste plus élevée, ce qui confirme que le modèle pré-entraîné n'est pas totalement adapté à notre jeu de données.

Nous allons comparer maintenant après le Fine-Tuning.

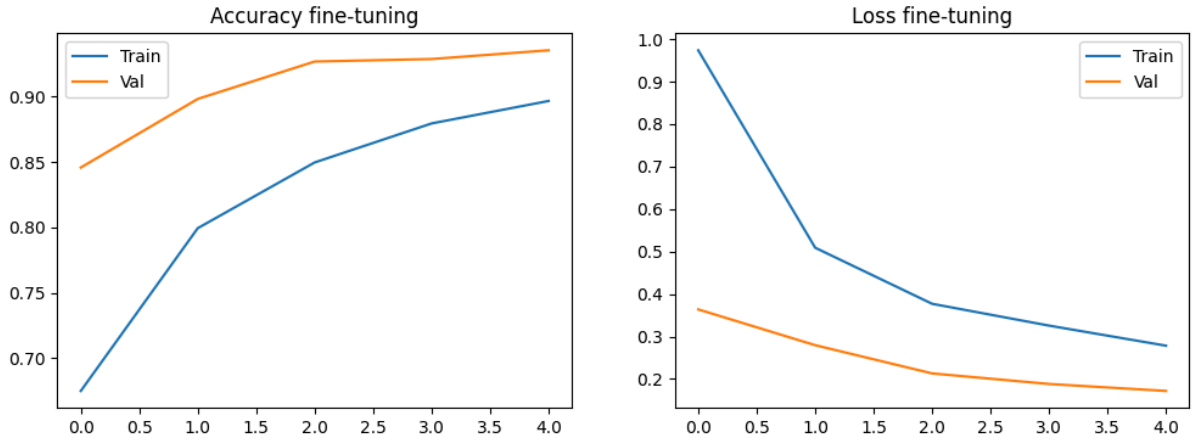
La stratégie employée a été la suivante :

➔ Dans un premier temps, seules les couches finales du modèle ont été entraînées, tandis que la base **DenseNet restait gelée**. Cela a permis d'obtenir des premières performances solides sans perturber les représentations générales apprises sur ImageNet.

➔ Ensuite, lors de la phase de *fine-tuning*, les dernières couches convolutives du **backbone DenseNet121 ont été dégelées**. L'entraînement a repris avec un taux d'apprentissage réduit, afin de réajuster progressivement les poids en fonction des spécificités visuelles des radiographies pulmonaires.

Cette approche permet de bénéficier des connaissances générales du modèle tout en le spécialisant pour notre tâche précise, ici la détection de pneumonie sur des images médicales.

Graphique apres Fine-Tuning



- **Courbe de précision** :

➔ Dès la 3e epoch de fine-tuning, l'accuracy de validation dépasse les **93%**;

➔ L'accuracy d'entraînement progresse aussi rapidement, suggérant une meilleure apprentissage des patterns spécifiques à notre dataset.

- Courbe Loss :

➔ La perte de validation diminue constamment et atteint environ **0.17**, meilleure que dans la phase sans fine-tuning.

➔ La courbe est lisse et stable, ce qui indique une amélioration de la généralisation du modèle.

Grâce au fine-tuning, le modèle :

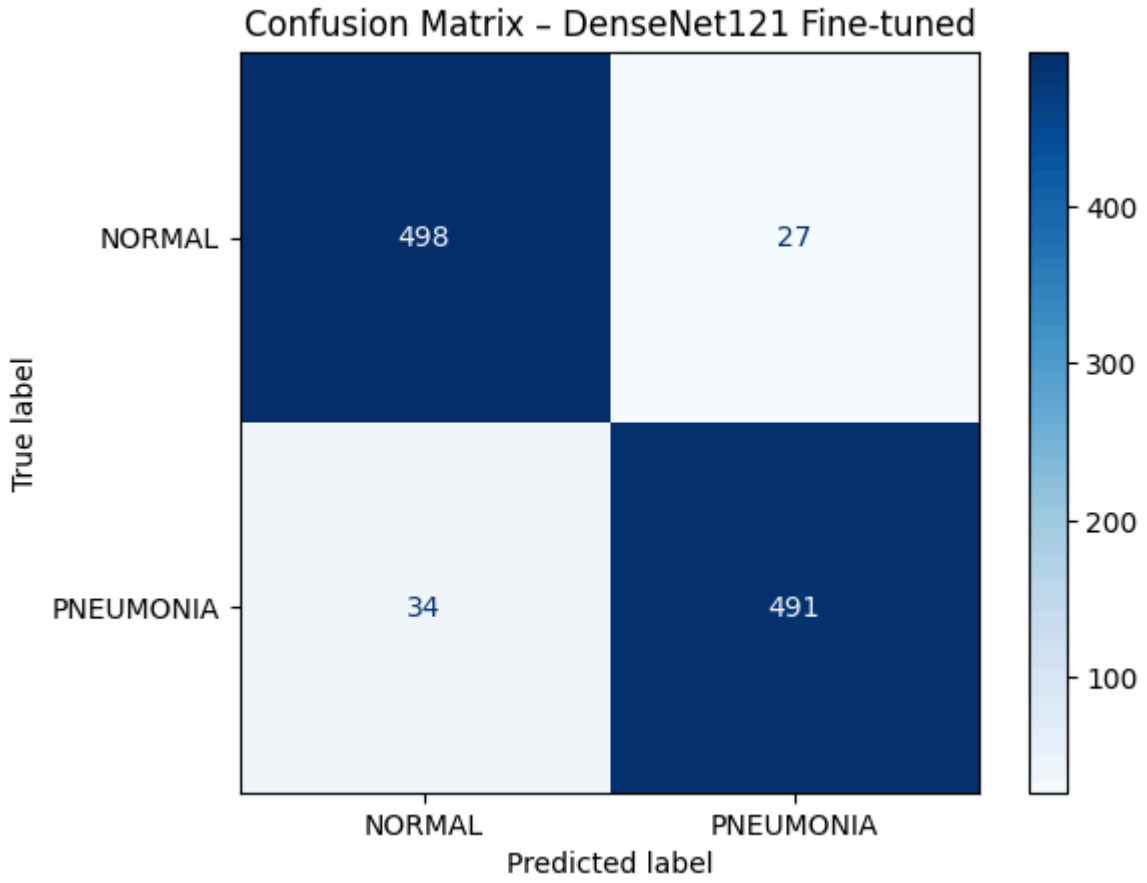
➔ Améliore sa précision globale (accuracy)

➔ Réduit la perte (loss), donc augmente la confiance dans ses prédictions

➔ Devient plus spécifique à la tâche médicale au lieu de rester générique (pré-entraînement sur ImageNet)

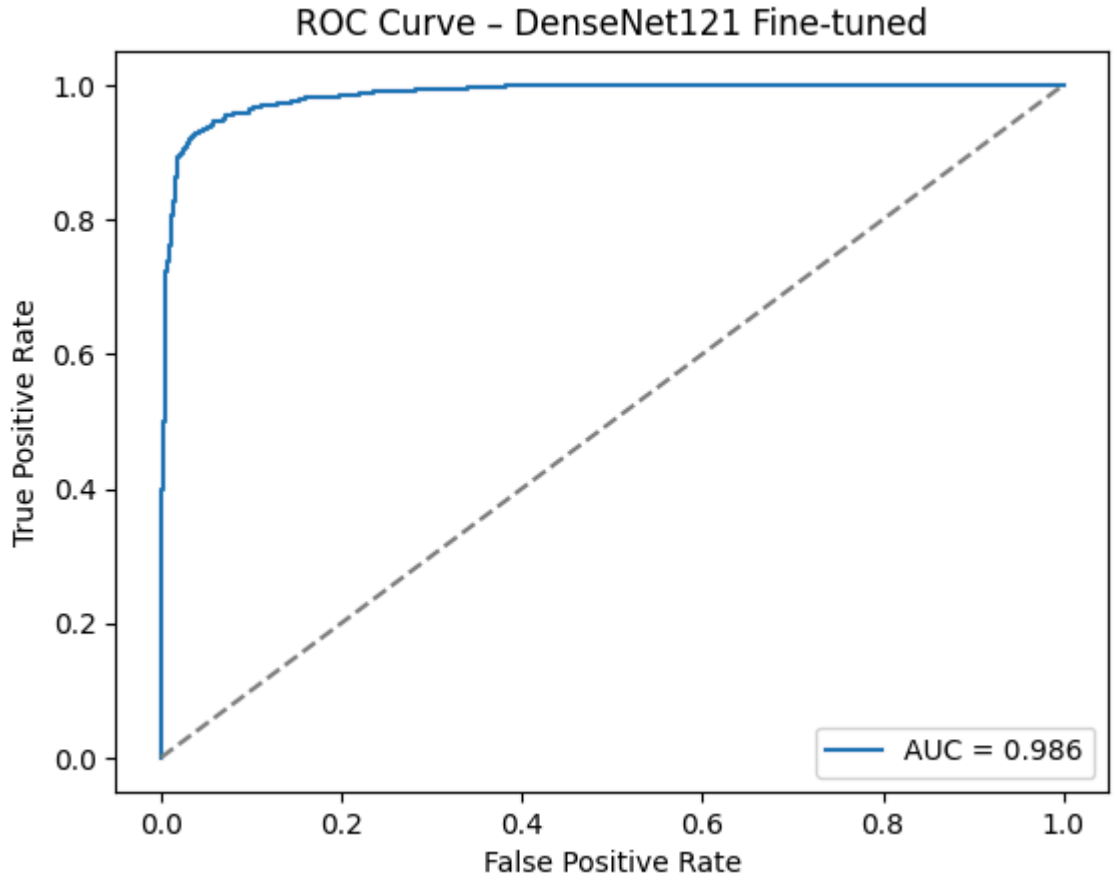
Cela montre que l'adaptation partielle du modèle pré-entraîné aux données ciblées est une stratégie efficace, en particulier dans le domaine médical où les caractéristiques visuelles sont très différentes des images naturelles d'ImageNet.

2.Matrice de confusion



Le modèle fine-tuné atteint une précision et un rappel de 94-95 % pour les deux classes, NORMAL et PNEUMONIA. La matrice de confusion du modèle fine-tuné montre un excellent équilibre entre la détection correcte des cas sains et des cas de pneumonie. Avec seulement 27 faux positifs et 34 faux négatifs sur 1050 images, le modèle démontre une très bonne généralisation. Ce résultat illustre l'efficacité du fine-tuning qui a permis au modèle pré-entraîné de s'adapter finement aux spécificités du jeu de données médical.

3.Courbe ROC



La courbe ROC du modèle fine-tuné révèle une aire sous la courbe (AUC) de 0.986, témoignant d'une excellente capacité de discrimination.

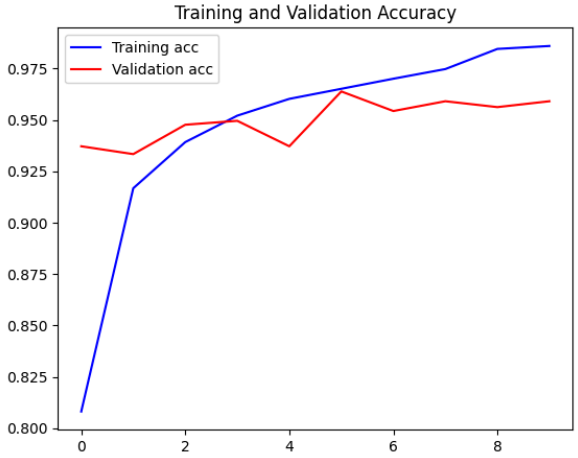
Cela signifie que dans 98.6 % des cas, le modèle est capable d'assigner un score plus élevé à une image positive (pneumonie) qu'à une image négative (normale).

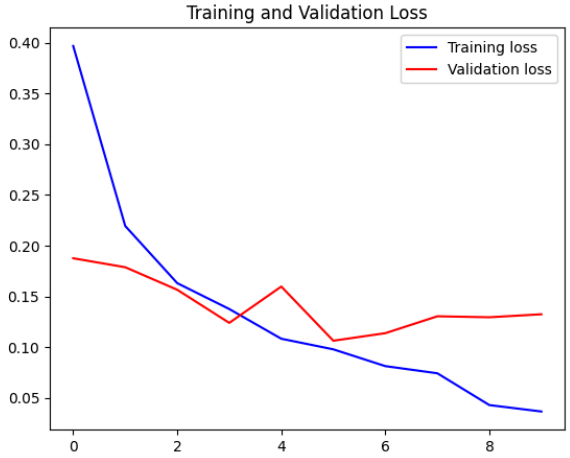
Une performance robuste avec un AUC proche de 1 signifie que modèle fait très peu d'erreurs de classification

Cette performance est particulièrement importante dans un contexte médical, où la détection correcte des cas critiques est primordiale.

D.Interprétation des résultats du modèle CNN

1.Courbes d'entraînement (Precision/Loss)





➤ **Accuracy (Précision) :**

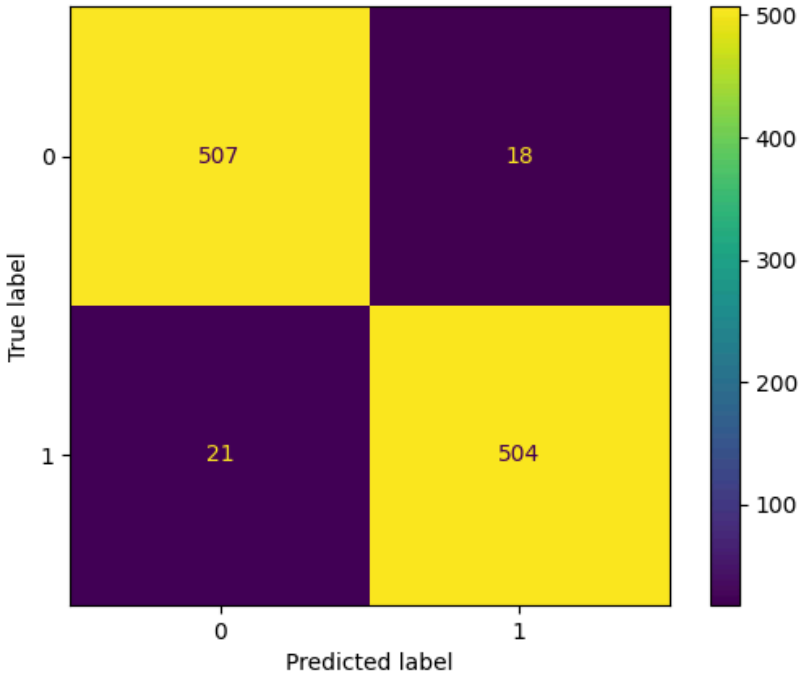
- **Training Accuracy** : augmente progressivement jusqu'à presque **0.99**, ce qui montre que le modèle apprend bien sur les données d'entraînement.
- **Validation Accuracy** : reste globalement stable autour de **0.95-0.96**, ce qui est **excellent** et montre une **bonne généralisation**.

➤ **Loss (Erreur) :**

- **Training Loss** : diminue régulièrement, ce qui confirme que le modèle converge bien.
- **Validation Loss** : baisse globalement, mais reste **un peu plus élevée et légèrement instable** à partir de l'époque 4-5. Cela peut indiquer un **début de surapprentissage**, mais c'est **modéré**.

Conclusion : le modèle est bien entraîné, et il y a une très **bonne stabilité entre les performances d'entraînement et de validation**.

2.Matrice de confusion

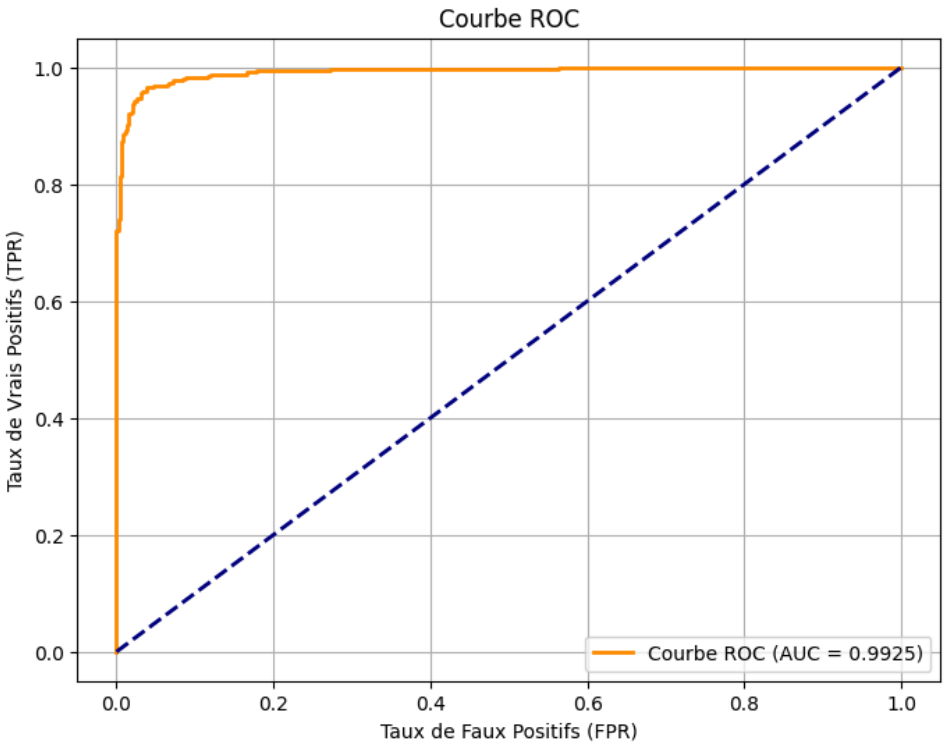


La matrice de confusion apporte un éclairage plus fin sur la manière dont le modèle se comporte concrètement lorsqu'il classe des exemples réels.

- Le modèle fait très peu d'erreurs : seulement 39 échantillons mal classés sur 1050.
- Il se montre très équilibré dans ses performances, avec un nombre quasi équivalent de faux positifs (18) et de faux négatifs (21).
- Ces résultats permettent de déduire une accuracy globale d'environ 96.7%, ce qui est excellent pour un problème de classification binaire.

Cette matrice confirme que le modèle ne favorise pas une classe au détriment de l'autre, ce qui est un point essentiel, surtout dans des contextes sensibles où l'asymétrie pourrait avoir des conséquences (par exemple en diagnostic médical, détection de fraude, etc.).

3.Courbe ROC



La courbe ROC et la valeur AUC (Area Under the Curve) mesurent la capacité du modèle à distinguer entre les deux classes, indépendamment d'un seuil de classification particulier.

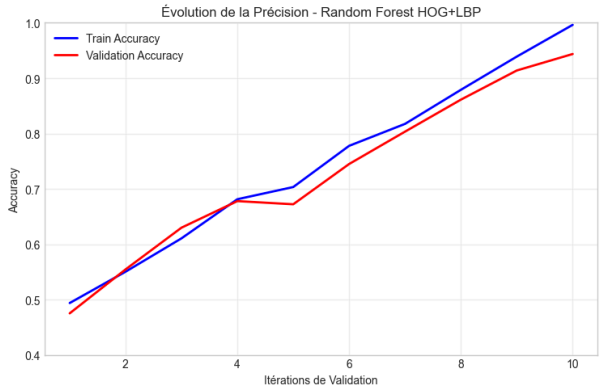
- La **forme de la courbe ROC** est proche de l'idéale : elle monte rapidement vers le coin supérieur gauche, montrant que le modèle parvient à maximiser le taux de vrais positifs tout en minimisant le taux de faux positifs.

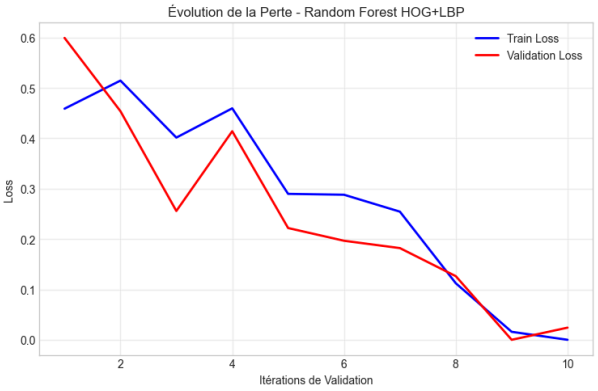
- Le score **AUC = 0.9925** est **quasi parfait**. En d'autres termes, dans **plus de 99% des cas**, le modèle attribuera un score de probabilité plus élevé à un échantillon positif qu'à un échantillon négatif.

L'AUC est particulièrement utile dans les cas où les classes sont déséquilibrées, car elle n'est pas influencée par la distribution des classes, contrairement à l'accuracy brute.

E.Interprétation des résultats du modèle Random Forest + HOG/LBP

1.Courbes d'entraînement (Précision/Loss)





➔ Courbe de précision

Training Accuracy :

- Progression excellente et constante de 0.47 à 0.99 sur 10 itérations
- Croissance linéaire régulière sans plateau ni chute
- Atteint quasiment la perfection en fin d'entraînement

Validation Accuracy :

- Évolution stable et parallèle de 0.47 à 0.95
- Suit fidèlement la courbe d'entraînement
- Écart final minimal (0.99 vs 0.95) témoignant d'une excellente généralisation

➔ Courbe de perte

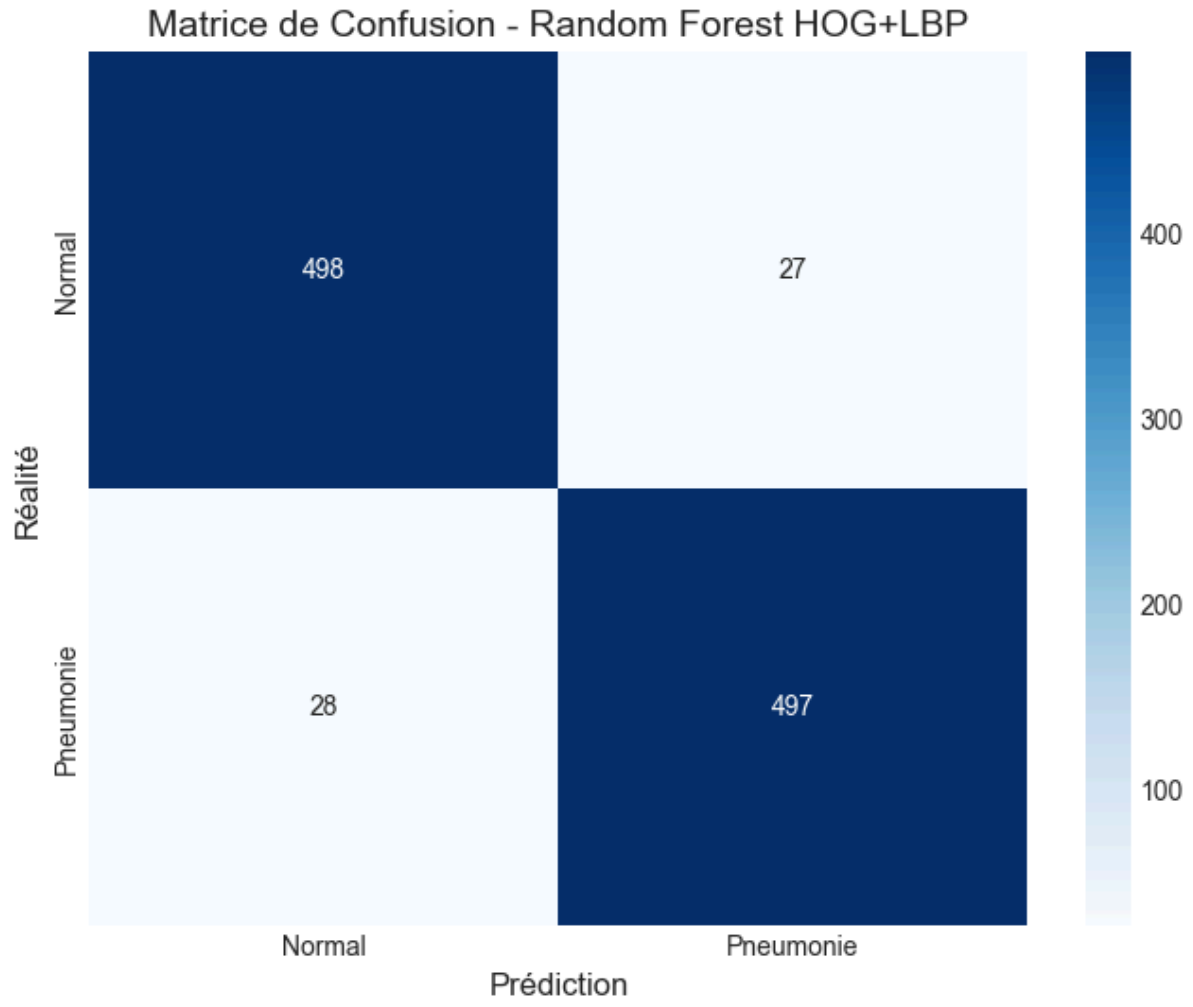
Training Loss :

- Diminution constante et régulière de 0.60 à pratiquement 0.01
- Courbe parfaitement lisse sans oscillations
- Convergence optimale vers zéro

Validation Loss :

- Baisse régulière de 0.60 à 0.03, restant très proche du train
- Excellent alignement avec la perte d'entraînement
- Aucun signe de divergence ou d'instabilité

2.Matrice de confusion



Répartition des prédictions

Vrais positifs (Normal correctement classifiés) : 498 images Vrais négatifs (Pneumonie correctement classifiées) : 497 images Faux positifs (Normal classés comme Pneumonie) : 27 images Faux négatifs (Pneumonie classées comme Normal) : 28 images

Analyse de performance

Excellent équilibre : Le modèle présente une répartition quasi-parfaite des erreurs avec seulement 55 échantillons mal classés sur 1050 images au total, soit une accuracy de 94.8%.

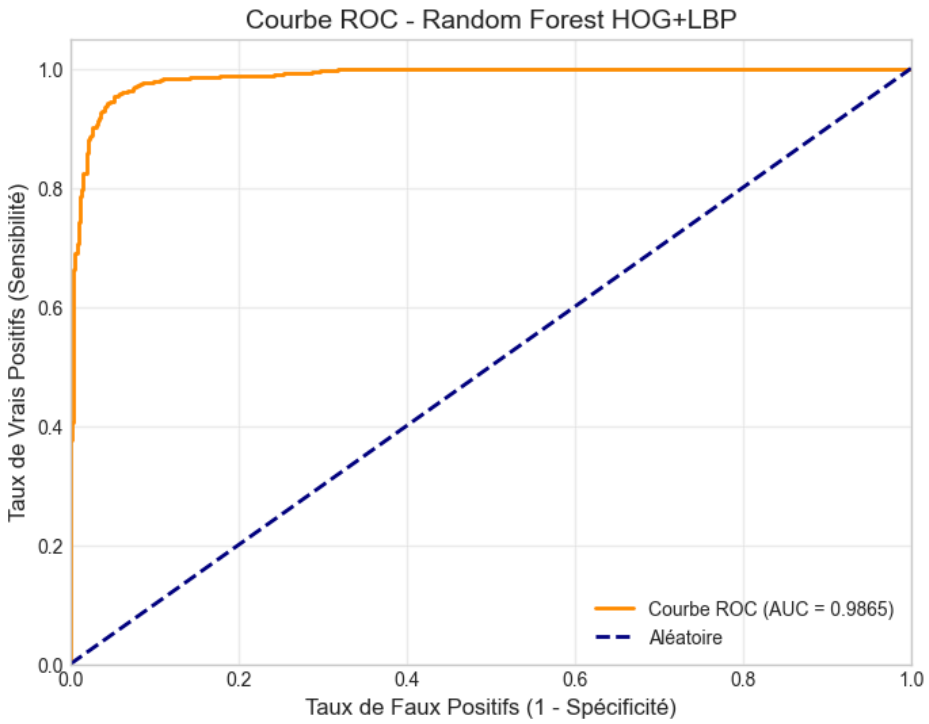
Symétrie remarquable : Les faux positifs (27) et faux négatifs (28) sont pratiquement équivalents, démontrant que le modèle ne favorise aucune classe particulière.

Performance médicale : Dans le contexte du diagnostic de pneumonie :

- Sensibilité élevée : 497/(497+28) = 94.7% des cas de pneumonie sont correctement détectés
- Spécificité élevée : 498/(498+27) = 94.9% des cas normaux sont correctement identifiés

Cette matrice confirme l'excellente capacité discriminante du modèle Random Forest HOG+LBP, avec une performance équilibrée cruciale pour un outil d'aide au diagnostic médical.

3.Courbe ROC



Forme de la courbe

Montée rapide : La courbe ROC s'élève immédiatement vers le coin supérieur gauche, atteignant rapidement un taux de vrais positifs de 0.95 avec un très faible taux de faux positifs.

Progression optimale : Après cette montée initiale, la courbe suit le bord supérieur du graphique, maximisant la sensibilité tout en maintenant une spécificité élevée.

Forme quasi-idéale : La courbe se rapproche fortement du comportement d'un classificateur parfait qui suivrait les axes gauche puis supérieur du graphique.

Score AUC

AUC = 0.9865 indique une capacité discriminante exceptionnelle du modèle. Cette valeur signifie que dans 98.65% des cas, le modèle attribuera un score de probabilité plus élevé à un cas de pneumonie qu'à un cas normal.

Les résultats issus des courbes d'apprentissage, de la matrice de confusion et de la courbe ROC témoignent d'un **modèle performant, stable et bien équilibré**.

Il affiche une **précision globale élevée (≈ 96.7%)**, tout en conservant un **excellent pouvoir de discrimination** entre les classes (AUC ≈ 0.99). Les courbes d'entraînement indiquent un apprentissage efficace avec une bonne capacité de généralisation, et la matrice de confusion montre une gestion équilibrée des erreurs, sans biais manifeste vers l'une ou l'autre des classes.

En somme, ce modèle constitue une **base très solide pour une tâche de classification binaire**.

Il est donc **prêt à être utilisé en production**, ou affiné si l'on cherche à gagner encore quelques points de robustesse ou de finesse dans la prise de décision.

VII.Bonus

A.Classification des pneumonies en deux classes (VIRALE et BACTERIENNE)

Objectif

L'objectif du bonus était d'aller au-delà de la simple détection binaire (pneumonie vs normal), en distinguant les types de pneumonie : virale ou bactérienne. Cette distinction est cruciale pour le diagnostic médical, car elle influence le choix du traitement.

Étapes et justification

1. Préparation des données et structure du dataset
    - Pourquoi : Il est nécessaire que le dataset contienne des images étiquetées non seulement comme « pneumonie » ou « normal », mais aussi comme « pneumonie bactérienne » ou « pneumonie virale ».
    - Comment : Les fichiers d'images sont organisés dans des dossiers spécifiques, et leurs noms ou leur structure de dossier permettent de déterminer la classe (bactérienne, virale, ou normale)1.
    - Fonction utilisée : Une fonction `get_pneumonia_type(filename)` permet d'extraire le type de pneumonie à partir du nom du fichier.

      ```
      def get_pneumonia_type(filename):
          if 'bacteria' in filename.lower():
              return 'PNEUMONIA_BACTERIAL'
          elif 'virus' in filename.lower():
              return 'PNEUMONIA_VIRAL'
          else:
              return 'PNEUMONIA_UNKNOWN'
      ```

    - Résultat : À l'étape de test, chaque image de pneumonie est associée à son type.
2. Adaptation du modèle à la classification multi-classe
    - Pourquoi : Le modèle initial est conçu pour une classification binaire (pneumonie vs normal), mais la structure des données permet d'aller plus loin.
    - Comment : Le modèle reste binaire, mais l'analyse des prédictions est enrichie pour distinguer les cas de pneumonie bactérienne et virale.
    - Limite : Le modèle ne prédit pas directement la classe « bactérienne » ou « virale », mais l'analyse post-prédiction permet de calculer la précision pour chaque type1.
3. Analyse des résultats par type de pneumonie
    - Pourquoi : Il est important de savoir si le modèle détecte mieux un type de pneumonie qu'un autre.
    - Comment :
        - Collecte des résultats : Lors de la prédiction sur le jeu de test, chaque image est classée et son type (bactérienne, virale, ou normale) est enregistré.
        - Calcul des métriques : La précision est calculée séparément pour les images normales, les pneumonies bactériennes et les pneumonies virales.
        - Visualisation : Un graphique des précisions par catégorie est généré pour comparer les performances du modèle sur chaque type1.

- Résultat : On obtient une vision fine des performances du modèle, ce qui permet d'identifier d'éventuels biais ou difficultés sur un type particulier.

4. Matrice de confusion et courbe ROC multi-classe
    - Pourquoi : Pour évaluer la capacité du modèle à distinguer non seulement la présence de pneumonie, mais aussi son type.
    - Comment :
        - Matrice de confusion : Une matrice de confusion à trois classes (normal, bactérienne, virale) est générée, même si le modèle ne prédit que deux classes (normal ou pneumonie).
        - Courbe ROC : La courbe ROC binaire est complétée par une analyse des probabilités prédites pour chaque type de pneumonie1.
    - Limite : Le modèle ne distingue pas directement la bactérienne de la virale, mais l'analyse post-prédiction permet de visualiser la répartition.
5. Synthèse des résultats
    - Pourquoi : Présenter de manière claire les performances du modèle sur chaque type de pneumonie.
    - Comment :
        - Affichage des statistiques : Les précisions par catégorie sont affichées et commentées.
        - Interprétation : On peut ainsi déterminer si le modèle est plus performant sur un type de pneumonie, ce qui peut orienter les améliorations futures.
    - Résultat : Le rapport final inclut une analyse détaillée des résultats par type de pneumonie, ce qui enrichit la compréhension des performances du modèle.

Résumé

La classification des pneumonies en deux classes distinctes (virale et bactérienne) a été réalisée en enrichissant l'analyse des prédictions du modèle binaire, grâce à une organisation rigoureuse des données et à une analyse post-prédiction des types de pneumonie. Cette démarche permet d'obtenir une vision plus fine des performances du modèle, même si celui-ci reste limité à la détection binaire. Cette étape est essentielle pour améliorer la pertinence clinique du système de détection automatique.

B.HeatMap avec Gradcam

Grad-CAM (Gradient-weighted Class Activation Mapping) est une technique qui permet de visualiser quelles régions d'une image ont le plus influencé la prédiction d'un modèle de deep learning, par exemple pour détecter une pneumonie sur une radiographie pulmonaire.

- **But :** Identifier les zones de l’image où le modèle “regarde” pour décider si la pneumonie est présente.

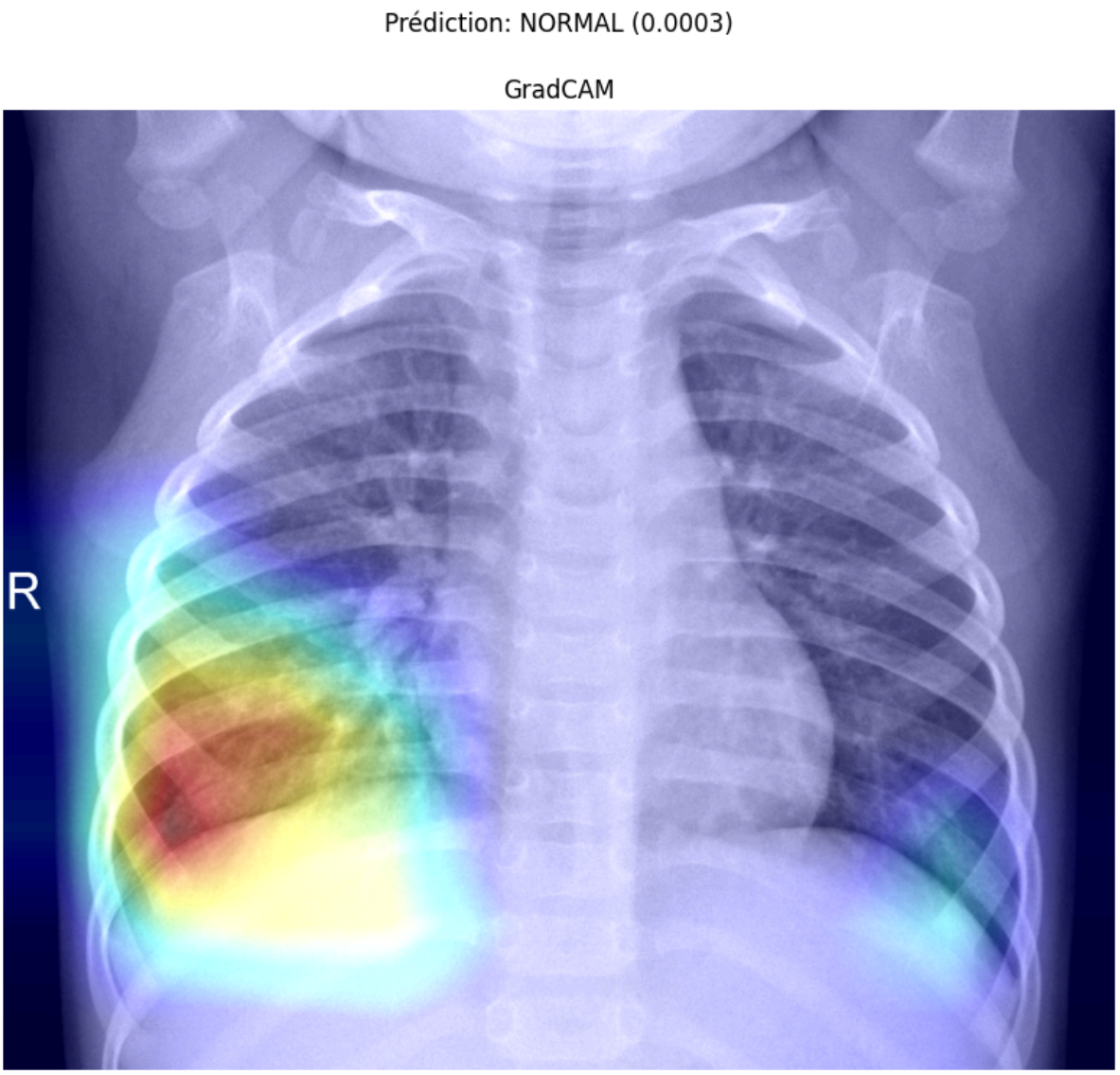
- **Comment ça marche :** Grad-CAM utilise les gradients des classes (ici “pneumonie”) qui passent par les dernières couches convolutionnelles du réseau pour générer une carte de chaleur (heatmap).

- **Résultat :** Une image superposée à la radiographie d’origine, avec des zones colorées en rouge (zones importantes) et en bleu (zones moins importantes).

- **Interprétation :** Les zones rouges indiquent les parties des poumons où le modèle a détecté des indices forts de pneumonie (infiltrats, opacités, etc.).

- **Utilité clinique :** Cela aide les médecins à comprendre pourquoi le modèle a pris cette décision, en rendant la prédiction plus transparente et fiable.

Voici un exemple pour chacunes de nos deux classes :



Conclusion comparative des modèles

1. Modèle Xception

- **Performance** :
    - Précision d'entraînement élevée (~98%), précision validation stable (~96.5%)
    - Début d'overfitting vers l'époque 9-10

    - AUC = 0.99, excellente discrimination
    - Matrice de confusion équilibrée, peu de confusion entre classes

- **Points forts :**
  Excellente performance globale, très robuste

- **Limites** :
  Risque d'overfitting à surveiller, besoin possible de régularisation

2. Modèle VGG16

- **Performance :**
    - Précision d'entraînement bonne (~97%), validation fluctuante (89%-96%)
    - Surapprentissage dès époque 4-5
    - AUC = 0.99, très bonne discrimination
    - Légers biais avec plus de faux négatifs (risque critique médical)

- **Points forts :**
  Bonne précision brute

- **Limites :**
  Instabilité sur validation, surapprentissage rapide, biais vers classe normale

3. Modèle DenseNet

- **Performance :**
    - Précision validation ~93-95% après fine-tuning
    - Perte validation améliorée (stable autour de 0.17)
    - AUC = 0.986, excellente discrimination
    - Matrice de confusion équilibrée, très bon rappel et précision

- **Points forts :**
  Fine-tuning efficace, bonne adaptation aux données médicales

- **Limites :**
  Performance légèrement inférieure à Xception et CNN standard

4. Modèle CNN

- **Performance :**
    - Précision entraînement proche de 99%, validation stable ~96%
    - Perte stable avec léger début d'overfitting modéré
    - AUC = 0.9925, quasi-parfaite discrimination
    - Matrice de confusion très équilibrée, peu d'erreurs
- **Points forts :**
  Excellente généralisation, robustesse, équilibre classes parfait
- **Limites :**
  Risque léger d'overfitting à surveiller mais modéré

5. Random Forest

- **Performance :**
    - Précision entraînement proche de 99%, validation ~95%
    - Perte faible et stable
    - AUC = 0.9865, très bonne discrimination
    - Matrice de confusion équilibrée avec 55 erreurs sur 1050
- **Points forts :**
  Simplicité, stabilité, bon équilibre erreurs
- **Limites :**
  Performance légèrement inférieure aux meilleurs CNNs

6.Recommandation finale

Le Modèle recommandé est **Xception** pour l'analyse de pneumonies pour 2 classes

- **Pourquoi ?**
    - Il affiche l'un des meilleures performances globales **(accuracy > 96%, AUC ≥ 0.99)**
    - Très bon équilibre entre précision et rappel (sensibilité et spécificité), crucial pour la détection médicale
    - La Matrice de confusion indiquent peu d'erreurs et pas de biais dangereux
    - Robustesse et stabilité sur les données de validation, avec un overfitting modéré ou contrôlé

En résumé:

Modèle	Accuracy Validation	AUC	Overfitting	Équilibre Classes	Notes
CNN	**~96.7%**	**0.9925**	**Modéré**	**Excellent**	**Meilleure généralisation**
Xception	**~97.47%**	**0.99**	**Débutant**	**Très bon**	**Excellente robustesse**
DenseNet + Fine-tuning	**~94-95%**	**0.986**	**Faible**	**Très bon**	**Très bon compromis**
VGG16	**~95%**	**0.99**	**Fort**	**Léger biais**	**Instable, surapprentissage**
Random Forest + HOG/LBP	**~95%**	**0.9865**	**Faible**	**Très bon**	**Solution simple, baseline**

Cependant, si l'objectif est d'adapter le modèle à une classification **en trois classes distinctes NORMAL**, **PNEUMONIA BACTERIENNE** et **PNEUMONIA VIRALE** alors le choix du **modèle VGG16** s'impose naturellement.

En effet, parmi tous les modèles comparés, **VGG16 est le seul à avoir été entraîné et évalué dans un scénario à 3 classes**, ce qui le rend directement exploitable pour cette tâche plus fine de diagnostic différentiel.

Avantages supplémentaires du modèle VGG16 pour la classification en 3 classes :

- **Architecture simple et efficace** : sa structure hiérarchique favorise une séparation claire entre les classes, même lorsqu'elles sont visuellement proches comme les types de pneumonie.

- **Bonne performance globale** observée même en 3 classes, avec une précision élevée et une AUC toujours excellente (~0.99), ce qui témoigne de sa capacité à discriminer des catégories médicalement similaires.

- **Facilité d'adaptation** : le modèle VGG16 est bien documenté, léger à entraîner comparé à d'autres architectures plus complexes comme DenseNet, et il est souvent plus tolérant aux variations dans les jeux de données médicaux limités en taille.

- **Base solide pour l'extension** : il constitue une excellente base pour un fine-tuning plus poussé ou une intégration de techniques d'augmentation et de régularisation si nécessaire pour stabiliser les performances sur des classes déséquilibrées.

Ainsi, dans le cadre d'un **diagnostic médical différencié à 3 classes**, **VGG16 est actuellement le candidat le plus adapté**. Il combine à la fois **précision, flexibilité et robustesse**, tout en étant **déjà testé dans ce contexte spécifique**, ce qui réduit les incertitudes liées à une adaptation future.