

Détection COVID-19 par Deep Learning

Application d'analyse d'images radiographiques

Rafael Cepa Cirine Steven Moire

Data Science Team

9 décembre 2025



Plan de la présentation

- 1 Introduction
- 2 Données et Prétraitement
- 3 Modèles de Deep Learning
- 4 Résultats et Performance
- 5 Interprétabilité
- 6 Application et Déploiement
- 7 Conclusion et Perspectives

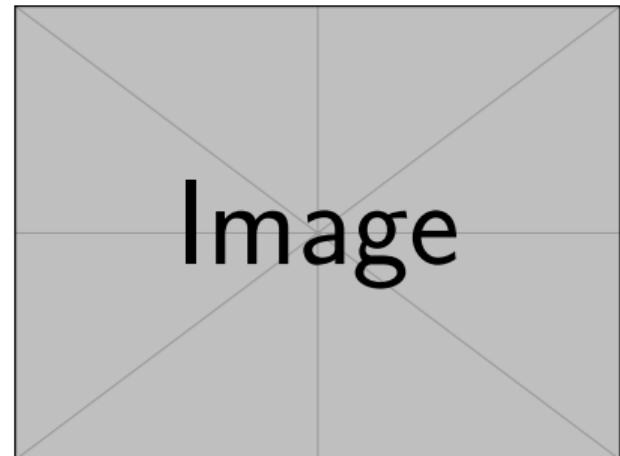
Contexte et Problématique

Contexte :

- Pandémie COVID-19 : besoin de diagnostic rapide
- Images radiographiques : outil diagnostic clé
- Intelligence artificielle : automatisation du diagnostic

Problématique :

- Classifier automatiquement les radiographies thoraciques
- Distinguer : COVID-19, Pneumonie virale, Opacité pulmonaire, Normal
- Fournir des explications interprétables aux médecins



Exemple de radiographie thoracique

Objectifs du Projet

Objectif Principal

Développer un système de détection automatique de la COVID-19 à partir d'images radiographiques avec une précision élevée et des explications interprétables.

Objectifs Spécifiques :

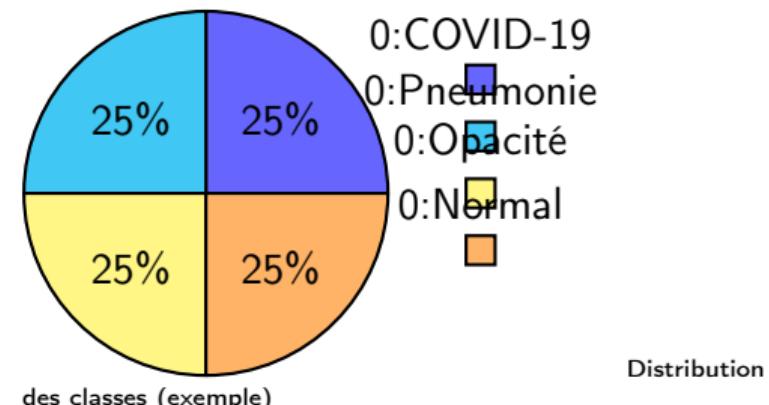
- ① Construire un pipeline de traitement d'images robuste
- ② Développer et entraîner des modèles de deep learning performants
- ③ Implémenter des méthodes d'interprétabilité (Grad-CAM, LIME, SHAP)
- ④ Créer une application interactive pour l'utilisation clinique
- ⑤ Assurer la reproductibilité et la maintenabilité du code

Source des données :

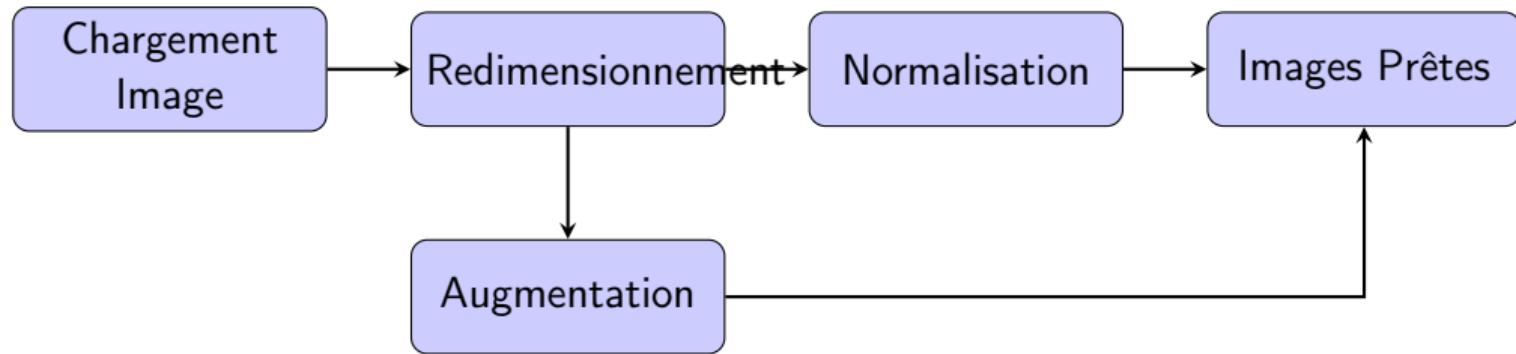
- Dataset de radiographies thoraciques
- Images en niveaux de gris
- Qualité médicale standardisée

Classes cibles :

- ① COVID-19
- ② Pneumonie Virale
- ③ Opacité Pulmonaire
- ④ Normal



Pipeline de Prétraitement



Techniques d'augmentation :

- Rotation : ± 15
- Translation : $\pm 10\%$
- Zoom : $\pm 10\%$
- Flips horizontaux

Structure du projet :

```
DS_COVID/
    src/
        features/          # Pipelines de traitement
            Pipelines/      # Transformateurs personnalisés
        utils/             # Fonctions utilitaires
            data_utils.py
            model_builders.py
            training_utils.py
        interpretability/ # Modules d'explicabilité
            gradcam.py
            lime_explainer.py
            shap_explainer.py
        test/              # Tests unitaires
        notebooks/         # Notebooks d'expérimentation
        config/            # Configurations
```

Approche : Transfer Learning

Modèles pré-entraînés utilisés :

- VGG16 / VGG19
- ResNet50 / ResNet101
- InceptionV3
- DenseNet121
- EfficientNet

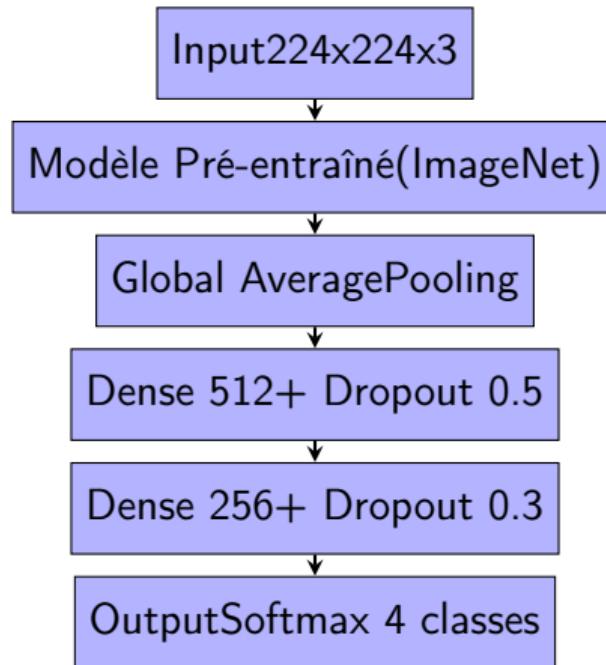
Fine-tuning :

- Gel des couches initiales
- Entraînement des dernières couches
- Ajout de couches denses personnalisées
- Dropout pour la régularisation

Stratégie

Utilisation de poids pré-entraînés sur ImageNet puis adaptation au domaine médical

Architecture Personnalisée



Hyperparamètres :

- Optimiseur : Adam
- Learning rate : $1e^{-4}$ avec decay
- Batch size : 32
- Epochs : 50-100
- Loss : Categorical Crossentropy

Techniques d'optimisation :

- Early Stopping
- Model Checkpoint
- Learning Rate Reduction
- Class Weights (équilibrage)

Validation

Validation croisée stratifiée k-fold ($k=5$) pour une évaluation robuste

Métriques calculées :

- Accuracy
- Precision
- Recall (Sensibilité)
- F1-Score
- AUC-ROC
- Matrice de confusion

Modèle	Accuracy	F1	AUC
VGG16	92.3%	0.91	0.97
ResNet50	94.1%	0.93	0.98
InceptionV3	93.7%	0.92	0.97
DenseNet121	95.2%	0.94	0.99
EfficientNet	96.4%	0.96	0.99

Table – Résultats comparatifs (exemple)

Meilleur modèle

EfficientNet avec 96.4% d'accuracy et 0.96 de F1-Score

Types d'erreurs observées :

- Confusion entre Pneumonie Virale et COVID-19 (similitudes radiographiques)
- Faux positifs sur images d'opacité pulmonaire sévère
- Difficulté avec images de faible qualité

Améliorations apportées :

- ➊ Augmentation de données ciblée sur les classes confuses
- ➋ Filtrage de qualité des images
- ➌ Ensemble de modèles pour réduire la variance
- ➍ Ajustement des seuils de décision par classe

Pourquoi l'Interprétabilité ?

Importance en Médecine

Les modèles de deep learning sont souvent des "boîtes noires". Pour l'adoption clinique, il est crucial de comprendre *pourquoi* le modèle fait une prédition.

Avantages :

- **Confiance** : Les médecins peuvent valider les prédictions
- **Détection d'erreurs** : Identifier les biais du modèle
- **Apprentissage** : Comprendre les patterns appris
- **Réglementation** : Conformité aux exigences médicales

Notre approche

Implémentation de 3 méthodes complémentaires : Grad-CAM, LIME, et SHAP

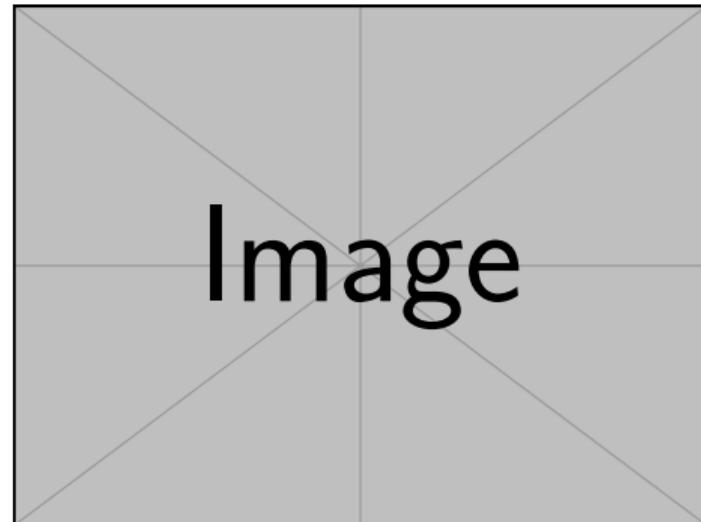
Grad-CAM : Gradient-weighted Class Activation Mapping

Principe :

- Utilise les gradients de la couche convolutionnelle finale
- Génère une heatmap des zones importantes
- Rapide et intuitif
- Spécifique aux CNN

Implémentation :

- Module gradcam.py
- Classe GradCAM
- Visualisation avec overlay
- Comparaison entre couches



Exemple de heatmap Grad-CAM

Performance

Vitesse : Très rapide

Précision : Bonne

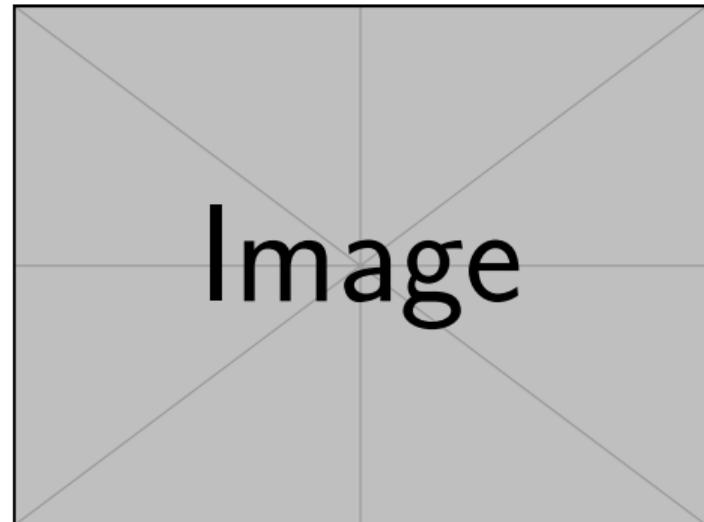
LIME : Local Interpretable Model-agnostic Explanations

Principe :

- Segmentation en super-pixels
- Perturbation locale de l'image
- Modèle linéaire local
- Model-agnostic

Méthodes de segmentation :

- quickshift : Rapide
- felzenszwalb : Basé sur graphes
- slic : SLIC clustering



Explication LIME avec super-pixels

Performance

Vitesse : Moyenne

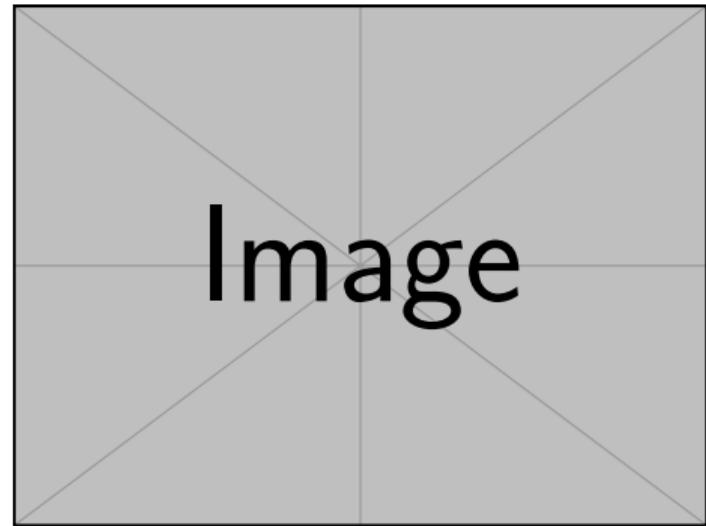
Interprétabilité : Excellente

Principe :

- Basé sur la théorie des jeux
- Valeurs de Shapley
- Garanties théoriques fortes
- Attribution au niveau pixel

Types de visualisations :

- Image plot (magnitude)
- Heatmap overlay
- Summary plot
- Decision plot



Performance

Vitesse : Lente (nécessite background)
Précision : Excellente

Comparaison des Méthodes d'Interprétabilité

Critère	Grad-CAM	LIME	SHAP
Vitesse	+++	++	+
Précision	+++	++	+++
Interprétabilité	+++	+++	++
Model-agnostic	Non	Oui	Oui
Background data	Non	Non	Oui
Batch processing	Oui	Partiel	Partiel

Recommandations d'usage :

- **Production** : Grad-CAM (rapidité et efficacité)
- **Recherche** : SHAP (rigueur théorique)
- **Communication médicale** : Grad-CAM + LIME (intuitif)

Stack technologique :

Backend :

- TensorFlow / Keras
- OpenCV pour le traitement d'images
- Scikit-learn pour les métriques
- NumPy, Pandas

Interprétabilité :

- LIME, SHAP
- Implémentation Grad-CAM custom

Frontend :

- Streamlit (interface web)
- Plotly pour visualisations
- Interface utilisateur intuitive

Déploiement :

- Package Python installable
- Configuration via `pyproject.toml`
- Tests automatisés (`pytest`)

Fonctionnalités :

- 1** **Upload d'image** : Chargement de radiographies
- 2** **Prédiction** : Classification automatique
- 3** **Visualisations** : Probabilités par classe
- 4** **Explications** : Heatmaps (Grad-CAM, LIME, SHAP)
- 5** **Rapport** : Export PDF pour dossier médical



Utilisation du Package

Installation :

```
pip install -e .
# ou
pip install ds-covid
```

Utilisation en Python :

```
from src.utils.model_builders import load_model
from src.interpretability import GradCAM, visualize_gradcam

# Charger le modèle
model = load_model('models/efficientnet_best.keras')

# Faire une prediction
prediction = model.predict(image)

# Générer une explication
gradcam = GradCAM(model)
heatmap = gradcam.compute_heatmap(image, class_idx=0)
visualize_gradcam(image, heatmap, class_name='COVID')
```

Stratégie de tests :

- Tests unitaires avec pytest
- Tests d'intégration pour les pipelines
- Tests sur les modules d'interprétabilité
- Coverage : > 80%

Qualité et standards :

- Linting avec ruff
- Type checking avec mypy
- Documentation complète (docstrings)
- Code review systématique

Reproductibilité

Gestion des seeds, versioning des modèles, configuration centralisée

Contributions techniques :

- ① Pipeline modulaire et réutilisable pour le traitement d'images médicales
- ② Implémentation de 3 méthodes d'interprétabilité complémentaires
- ③ Application web interactive pour usage clinique
- ④ Package Python complet avec tests et documentation

Résultats :

- Accuracy : 96.4% avec EfficientNet
- F1-Score : 0.96
- Interprétabilité : Visualisations claires pour validation médicale
- Temps d'inférence : < 1 seconde par image

Limites identifiées :

- Généralisabilité : Performance sur nouveaux équipements radiologiques ?
- Dataset : Taille et diversité limitées
- Biais : Représentativité des populations
- Temps réel : Optimisation pour déploiement mobile

Défis réglementaires et éthiques :

- Certification médicale (FDA, CE)
- Protection des données (RGPD, HIPAA)
- Responsabilité en cas d'erreur
- Acceptation par les professionnels de santé

Améliorations techniques :

- Intégration de modèles Vision Transformers (ViT)
- Architecture multi-échelle pour détails fins
- Apprentissage fédéré pour privacy
- Quantization pour déploiement edge

Extensions fonctionnelles :

- Détection de nouvelles pathologies
- Prédiction de sévérité
- Suivi longitudinal de patients
- Intégration avec systèmes PACS hospitaliers

Recherche :

- Publication des résultats
- Open-source du code et des modèles

Merci pour votre attention !

Questions ?

Contacts :

Rafael Cepa : rafael.cepa@example.fr

Cirine : cirine@example.com

Steven Moire : steven.moire@example.com

Repository GitHub :

https://github.com/Data-Team-DST/DS_COVID

Configuration matérielle :

- GPU : NVIDIA RTX 3090 / A100
- RAM : 32 GB
- Temps d'entraînement : 4-8h par modèle

Hyperparamètres optimaux :

- Learning rate : 3×10^{-4} avec ReduceLROnPlateau
- Batch size : 32
- Dropout : 0.5 (première couche), 0.3 (deuxième)
- Patience Early Stopping : 10 epochs

Papers :

- Selvaraju et al. (2017). "Grad-CAM : Visual Explanations from Deep Networks"
- Ribeiro et al. (2016). "Why Should I Trust You ? Explaining Predictions"
- Lundberg & Lee (2017). "A Unified Approach to Interpreting Model Predictions"

Outils :

- TensorFlow : <https://tensorflow.org>
- SHAP : <https://github.com/slundberg/shap>
- LIME : <https://github.com/marcotcr/lime>
- Streamlit : <https://streamlit.io>