T-DEV-810 - ZOIDBERG 2.0

Msc Pro

2023 - 2025

Auteurs :

Ayoub AIT-ABDELLAH,

Abraham DIAW,

Ian DUQUENNOI,

Yann OBAMA,

Jean-François TRAORE

2 juillet 2024

Aperçu des données

Commençons par un aperçu de nos données. Nous avons travaillé avec un ensemble d'images dont la taille moyenne originale était de ***1328 x 971*** pixels. Notre jeu de données se compose de deux parties principales :

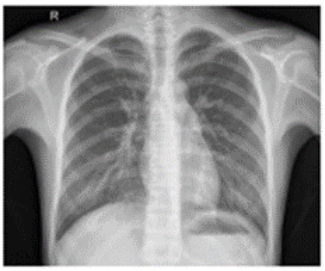
1. Un ensemble d'entraînement de **5216 images**, dont **3875** sont positives et **1341** négatives.
2. Un ensemble de test de **624 images**, avec **390** positives et **234** négatives. Cette répartition nous a permis d'entraîner nos modèles de manière robuste et de les tester sur des données non vues.

Prétraitement des images

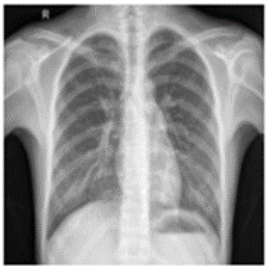
Avant de pouvoir utiliser ces images dans nos modèles, nous avons dû effectuer plusieurs étapes de prétraitement cruciales :

1. Redimensionnement : Toutes les images ont été redimensionnées à une taille uniforme de ***224 x 224*** pixels. Ce choix n'est pas anodin, c'est la taille d'entrée standard pour de nombreux réseaux de neurones convolutifs, dont le ResNet que nous avons utilisé.
2. Conversion en niveaux de gris : Cette étape a permis de réduire la complexité de nos données tout en conservant les informations essentielles pour notre tâche de classification.
3. Normalisation des pixels : Nous avons normalisé les valeurs des pixels pour les ramener dans une plage standard, ce qui aide à stabiliser l'entraînement de nos modèles.

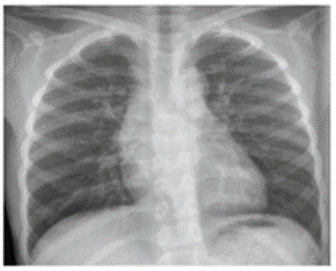
Original Size: (2498, 2057)

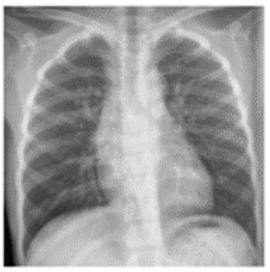


Resize : (224, 224)



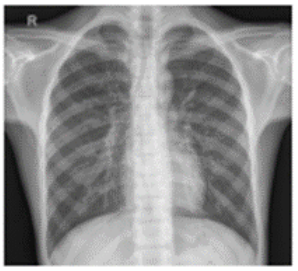
Original Size: (1663, 1326)

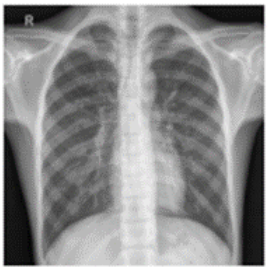




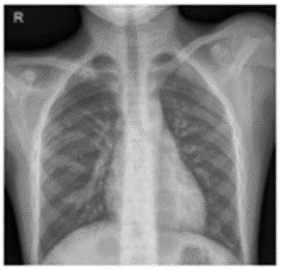
Original Size: (2111, 1509)

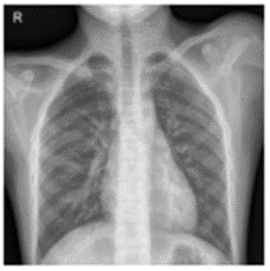
Original Size: (2031, 1837)

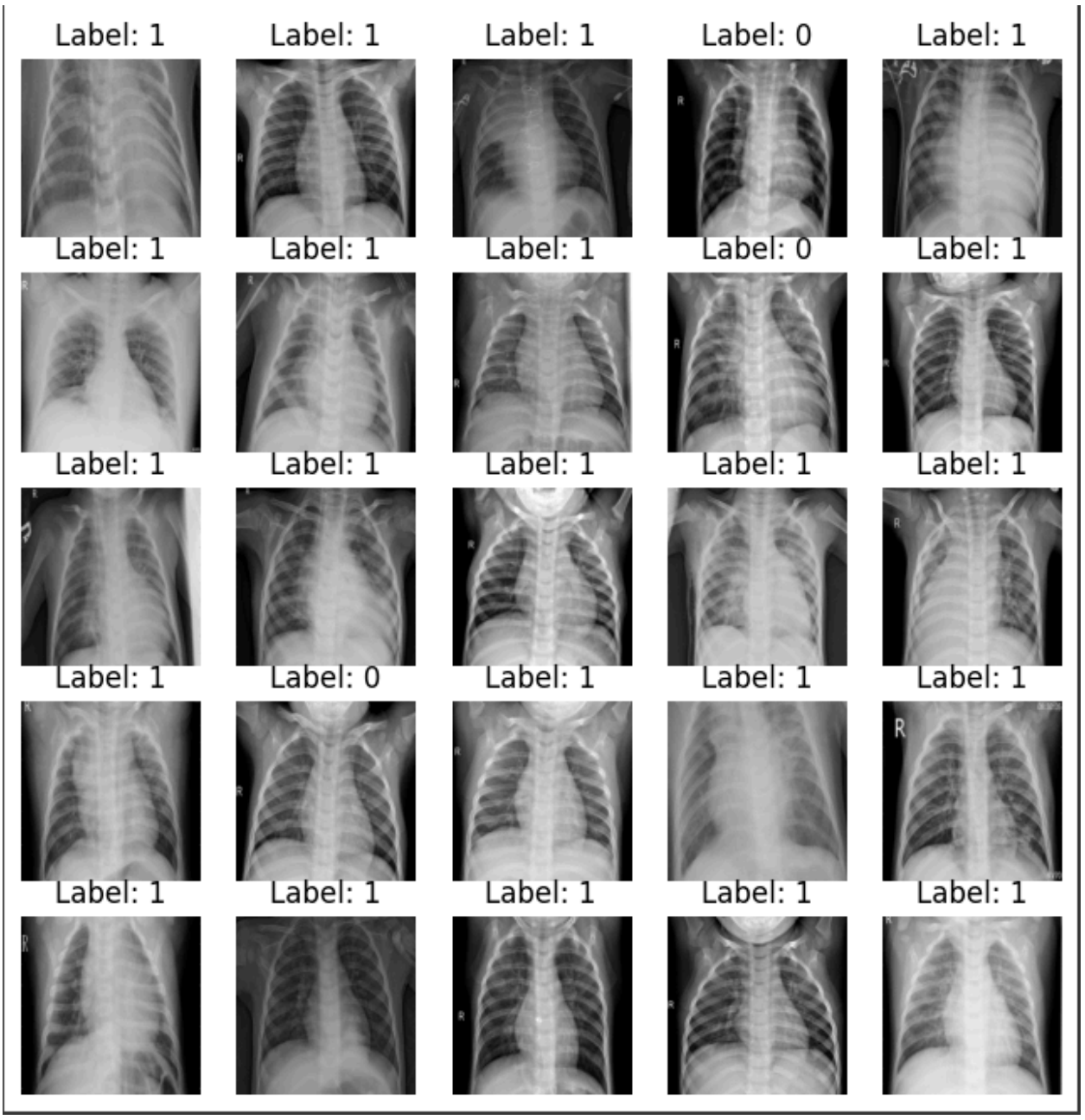




Original Size: (1863, 1785)







Régression logistique

Notre première approche a été d'utiliser la régression logistique, un modèle simple mais souvent efficace pour les tâches de classification binaire.

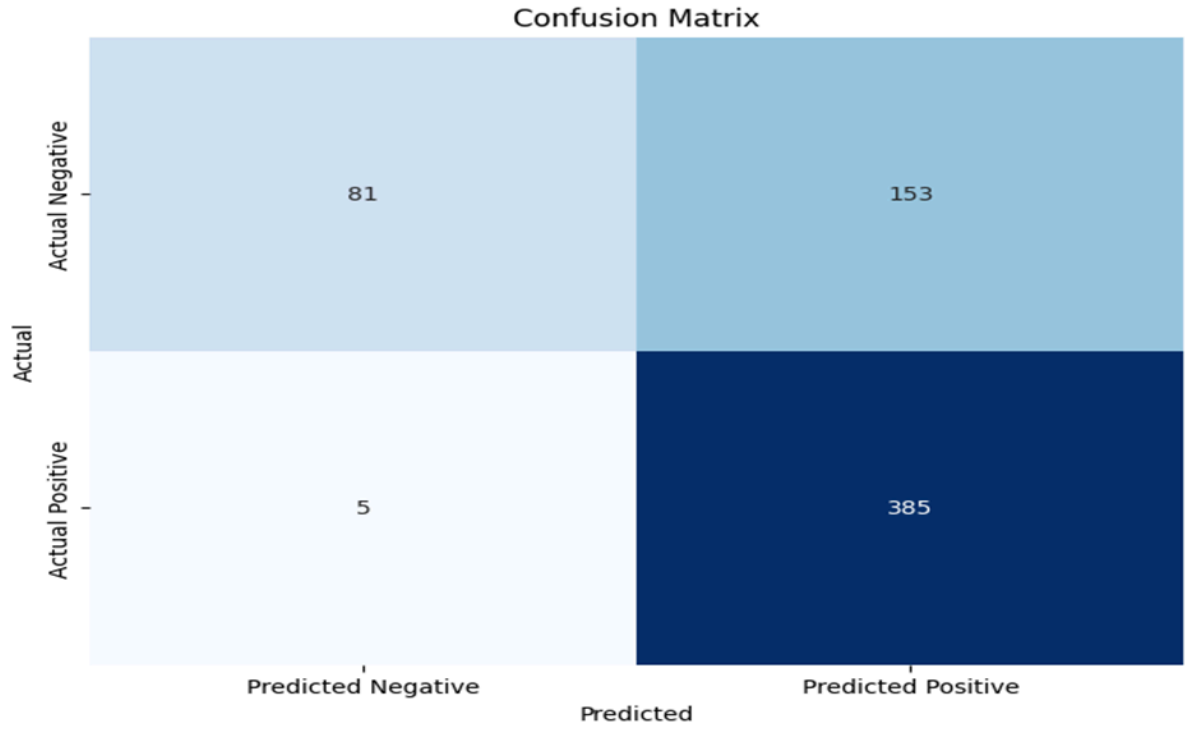
Après optimisation des hyperparamètres, nous avons obtenu les meilleurs résultats avec :

- C = 0.1, qui contrôle la force de régularisation
- max_iter = 600, limitant le nombre d'itérations
- solver = lbfgs, un algorithme d'optimisation efficace pour ce type de problème

Avec ces paramètres, nous avons atteint une précision de **74.67**% sur notre ensemble de test.

L'analyse de la matrice de confusion nous a montré que le modèle était particulièrement efficace pour détecter les cas positifs, mais avait plus de difficultés avec les cas négatifs. Cela nous a donné des pistes d'amélioration pour nos modèles suivants.

Voici les résultats complets dans une matrice de confusion :



Le modèle a bien détecté les positifs (385 bonnes prédictions contre 5 fausse) mais à pas très bien performé sur les négatifs (81 bonnes prédictions pour 153 fausse).

Arbre de décision

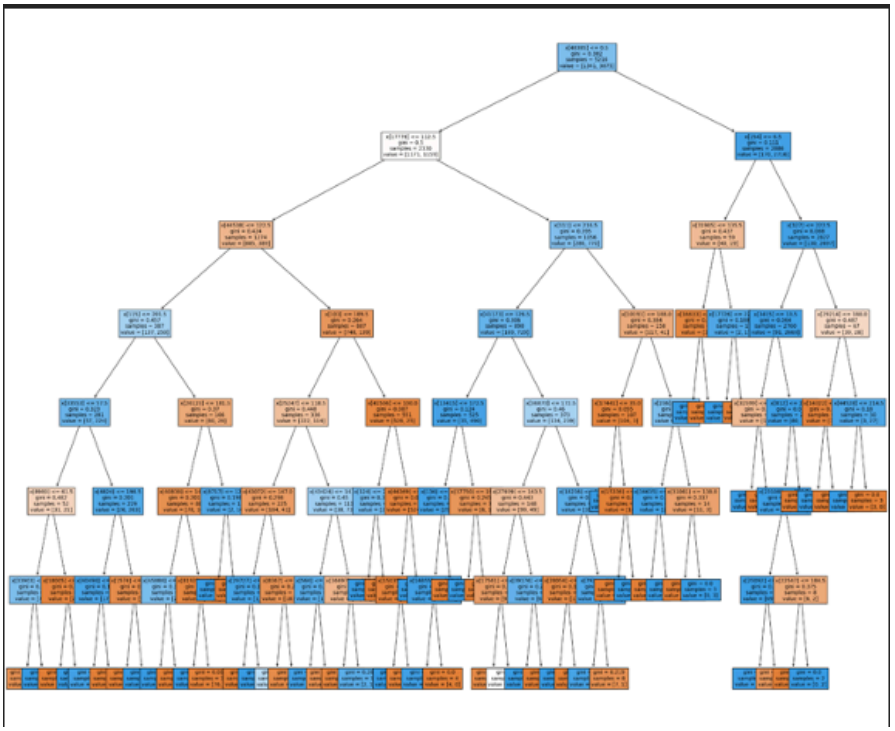
Notre deuxième approche a été l'utilisation d'un arbre de décision. Ce type de modèle a l'avantage d'être facilement interprétable, ce qui peut être crucial dans certains contextes d'application.

Après optimisation, nos meilleurs résultats ont été obtenus avec :

- Un critère d'entropie pour la division des nœuds
- Une profondeur maximale de **7** pour éviter le surapprentissage

Avec ces paramètres, nous avons atteint une précision de **73.36**%, légèrement supérieure à celle de la régression logistique.

Ce résultat proche nous a montré que la complexité additionnelle de l'arbre de décision n'apportait pas un gain significatif pour notre problème spécifique.



Réseau de neurones convolutif (CNN)

Notre approche finale, et la plus performante, a été l'utilisation d'un réseau de neurones convolutif.

Nous avons implémenté un premier modèle avec l'architecture présentée ci-dessous :

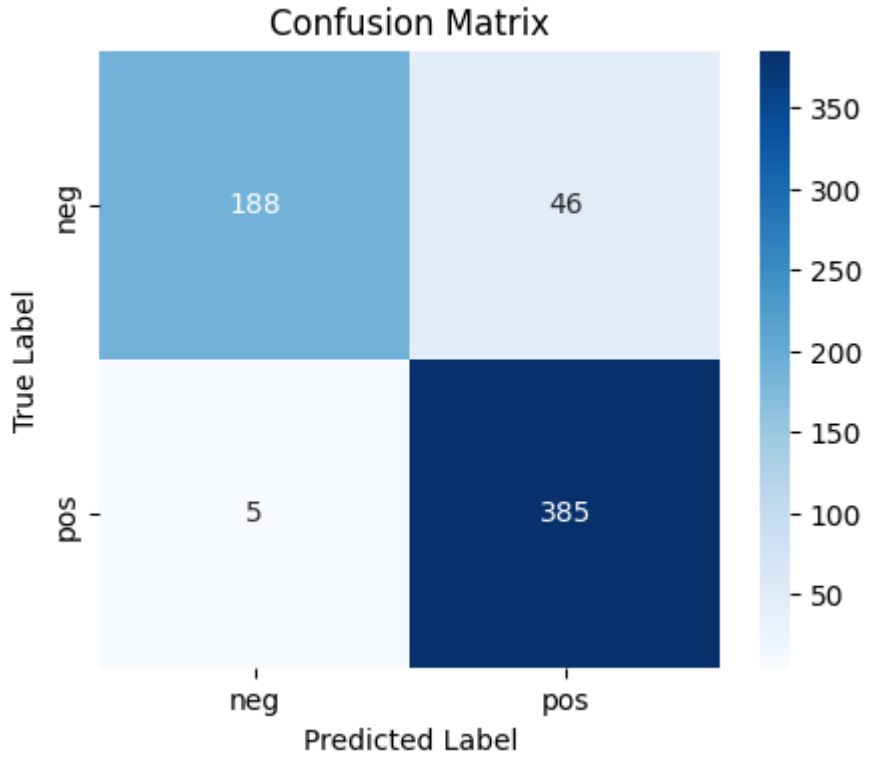
Couche	Type	Paramètres	Entrée	Sortie	Fonction
conv1	Conv2d	kernel_size=(5, 5), stride=(1, 1)	1 canal	6 canaux	Extraction de caractéristiques de base
pool	MaxPool2d	kernel_size=2, stride=2	6 canaux	6 canaux	Réduction de la dimensionnalité spatiale
conv2	Conv2d	kernel_size=(5, 5), stride=(1, 1)	6 canaux	16 canaux	Extraction de caractéristiques plus complexes

fc1	Linear	in_features=44944, out_features=120	44944	120	Aplatissement et réduction des caractéristiques
fc2	Linear	in_features=120, out_features=84	120	84	Poursuite de la réduction des caractéristiques
fc3	Linear	in_features=84, out_features=1	84	1	Sortie finale (classification ou régression)

Après un entraînement de 5 epoch, ce modèle à eu une

Nous avons utilisé une approche de transfer learning, en partant d'un modèle pré-entraîné sur ImageNet, que nous avons ensuite affiné sur notre jeu de données spécifique.

Le résultat a été spectaculaire, avec une précision finale de **91.83**% sur notre ensemble de test, surpassant largement nos deux approches précédentes.



On a des bons résultats pour détecter les positifs et on a une meilleure prédiction pour détecter les négatifs que les autres modèles donnant une bonne performance générale.

CNN vs Réseau de neurones classique

L'utilisation d'un CNN plutôt qu'un réseau de neurones classique (fully connected) a apporté plusieurs avantages cruciaux pour notre tâche de classification d'images :

1. Prise en compte de la structure spatiale : Les couches convolutives permettent au modèle de capturer des motifs locaux dans les images, ce qui est crucial pour la reconnaissance d'objets ou de caractéristiques.
2. Partage de paramètres : Les filtres convolutifs sont appliqués sur toute l'image, ce qui réduit considérablement le nombre de paramètres à apprendre par rapport à un réseau fully connected.

Ces avantages se sont traduits par une amélioration significative des performances. Alors qu'un réseau fully connected atteignait une précision de **78.36**%, notre CNN a atteint **91.83%**.

Parallélisation des données

Pour optimiser l'entraînement de nos modèles, particulièrement le CNN, nous avons utilisé le DataLoader de PyTorch pour paralléliser le chargement et le prétraitement des données.

Cette approche nous a permis :

1. D'accélérer l'entraînement.
2. D'optimiser l'utilisation des ressources : Nous avons pu maintenir notre GPU à une utilisation constante de près de 100% durant l'entraînement.

Benchmark des méthodes de classification

Model	Parameters	Accuracy (%)
Logistic Regression	C=1.0, solver='lbfgs', max_iter=100	71.56
Logistic Regression	C=0.1, solver='lbfgs', max_iter=600	74.67
Decision Tree	criterion='gini', max_depth=30	70.51
Decision Tree	criterion='entropy', max_depth=5	73.39
Neural Network	activation='ReLU', hidden_layers=7	78.36
CNN	activation='ReLU', convolutional_layers=2, epochs=5	79.50
CNN (ResNet18)	-	91.83

1. **Précision globale** : Le CNN (ResNet18) surpasse nettement les deux autres modèles avec une précision de 91.83%, contre 75% pour l'arbre de décision et 74.679% pour la régression logistique.
2. **Temps d'entraînement** : L'arbre de décision est généralement le plus rapide à entraîner, suivi de près par la régression logistique. Le CNN, en particulier un modèle complexe comme ResNet18, nécessite beaucoup plus de temps d'entraînement.
3. **Complexité du modèle** : La régression logistique est la plus simple, l'arbre de décision est de complexité moyenne, tandis que le CNN est le plus complexe.
4. **Interprétabilité** : L'arbre de décision offre la meilleure interprétabilité, suivi par la régression logistique. Le CNN est le moins interprétable des trois.
5. **Performance sur les classes** : Le CNN offre les meilleures performances tant sur les positifs que sur les négatifs. La régression logistique excelle sur les positifs mais est faible sur les négatifs.
6. **Capacité de généralisation** : Le CNN a généralement la meilleure capacité de généralisation, suivi par la régression logistique. Les arbres de décision peuvent être sujets au surapprentissage.
7. **Besoins en données et ressources** : Le CNN nécessite généralement plus de données et de ressources computationnelles que les deux autres modèles.
8. **Flexibilité** : Le CNN est le plus flexible et peut s'adapter à des tâches complexes de vision par ordinateur. La régression logistique est la moins flexible des trois.

En conclusion, chaque modèle a ses forces et ses faiblesses. Le choix dépend des priorités du projet : haute précision (CNN), rapidité et simplicité (régression logistique), ou interprétabilité (arbre de décision).