

DS50: Identification de tumeurs à l'aide de la Data Science et de modèles d'apprentissage

Eleanore RENAUD, Elise ALBRECHT, Theo GOUIN, Jérémie KIMENAU, Osman GAYGUSUZ, Gaspard ROCHU

Le 19 Juin 2024

Abstract

Le diagnostic de problèmes de santé comme des tumeurs cérébrales à l'aide de modèles en machine learning ou en deep learning est un domaine de recherche en plein essor. Dans ce projet, nous avons exploré différentes architectures de réseaux de neurones pour la classification des tumeurs cérébrales, notamment DenseNet, ResNet, EfficientNet et YOLO.

Keywords: *tumeurs cérébrales, IRM, machine learning, deep learning, réseaux de neurones, CNN*

1. Introduction

La détection précoce et précise des tumeurs est cruciale pour améliorer les taux de survie des patients et optimiser les traitements médicaux. Les tumeurs, caractérisées par une prolifération anormale des cellules, peuvent se former dans diverses parties du corps, y compris le cerveau. Les techniques d'imagerie telles que l'IRM et la tomodensitométrie (TDM) sont couramment utilisées pour détecter ces tumeurs. Cependant, l'interprétation manuelle de ces images par les radiologues est un processus complexe, sujet à des erreurs et dépend fortement de l'expertise du praticien.

Pour améliorer la fiabilité et la précision des diagnostics, les technologies d'intelligence artificielle (IA), notamment les réseaux de neurones convolutifs (CNN), montrent des performances prometteuses dans la reconnaissance des images médicales. Ces technologies, intégrant des architectures avancées telles que DenseNet, ResNet, EfficientNet et YOLO, permettent d'automatiser et d'affiner la détection des tumeurs à partir d'images IRM.

Le cerveau humain, centre de contrôle de toutes les fonctions corporelles, est un organe complexe et crucial. Constitué de divers types de tissus, incluant le liquide céphalo-rachidien, la substance blanche et la substance grise, il rend la détection des tumeurs cérébrales particulièrement difficile. Les tumeurs peuvent grandement affecter les fonctions cérébrales en raison de l'espace limité dans lequel elles se développent, rendant la planification des traitements et des interventions chirurgicales complexe et délicate.

Ce projet se concentre sur le développement et l'évaluation de modèles de CNN pré-entraînés pour améliorer la détection des tumeurs cérébrales. L'objectif est de déterminer la meilleure architecture en termes de performance et de robustesse, afin de contribuer de manière significative aux techniques de diagnostic médical automatisé. Ce rapport présente une analyse détaillée du travail réalisé et des améliorations potentielles envisagées pour renforcer la précision et la rapidité des diagnostics, offrant ainsi un soutien supplémentaire aux radiologues.

2. Méthodologie

2.1. Méthode Agile

Pour mener à bien ce projet, nous avons adopté une approche méthodologique basée sur la méthode agile, favorisant la collaboration, la flexibilité et l'amélioration continue.

Nous étions organisés en trois équipes de deux personnes, chaque binôme travaillant sur un modèle de CNN différent : DenseNet, ResNet, EfficientNet, etc. Notre objectif était de permettre à chaque binôme de se concentrer sur l'implémentation et l'optimisation de leur modèle respectif, puis de partager leurs résultats avec les autres équipes.

Les étapes de notre méthodologie agile comprenaient :

- Sprints courts: Nous avons travaillé en cycles de développement courts (sprints) de trois semaines, ce qui nous permettait de nous adapter rapidement aux retours et aux nouvelles idées tout en nous laissant le temps de bien implémenter le modèle.
- Réunions de synchronisation: À la fin de chaque sprint, nous organisions des réunions pour partager les progrès de chaque binôme, discuter des défis rencontrés et planifier les étapes suivantes. Ces réunions nous ont permis de rester alignés sur les objectifs du projet et de bénéficier des retours constructifs des autres équipes.
- Intégration continue: À chaque sprint, nous intégrions et comparions les résultats obtenus par les différents binômes. Cette intégration nous a permis d'identifier les points forts et les faiblesses de chaque modèle, et de déterminer les meilleures pratiques à adopter pour améliorer les performances globales.
- Amélioration continue: Grâce aux retours réguliers et à l'évaluation continue des performances des modèles, nous avons pu affiner nos approches, ajuster les hyperparamètres et améliorer les techniques de prétraitement des données. Cette approche nous a permis d'optimiser progressivement les modèles et d'assurer une montée en compétence collective.

En adoptant cette méthode de travail agile, nous avons pu maximiser notre productivité, favoriser la collaboration et garantir une évaluation rigoureuse et comparative des différents modèles de CNN appliqués à la détection des tumeurs cérébrales. Ce processus a non seulement amélioré la qualité de notre travail, mais a également renforcé notre capacité à innover et à nous adapter aux défis rencontrés au cours du projet.

2.2. Diagrammes de Gantt

Afin de réaliser au mieux notre projet nous avons réalisé un planning sous la forme de diagramme de Gantt, comprenant les éléments listés ci-dessus.

	Date Sémaine	18/3	25/3	1/4	8/4	15/4	22/4	29/4	6/5	13/5	20/5	27/5	3/6	10/6	17/6
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Réunion interne															
Sprint															
Point d'avancée avec Mr.Gaber															
Rédaction du rapport															

Figure 1. Diagramme de Gantt théorique pour le projet

	Date Sémaine	18/3	25/3	1/4	8/4	15/4	22/4	29/4	6/5	13/5	20/5	27/5	3/6	10/6	17/6
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Réunion interne															
Sprint															
Point d'avancée avec Mr.Gaber															
Rédaction du rapport															

Figure 2. Diagramme de Gantt réel du projet

La différence entre le diagramme théorique et réel vient du décalage du premier point d'avancé, ayant pour but de nous détailler

de façon techniques notre premier sprint. Ce retard a bloqué le reste des avancés. Nous avons pris la décision de décaler nos sprint d'une semaine.

3. Données utilisées

Pour ce projet, nous avons choisi d'utiliser le dataset suivant disponible sur Kaggle : [Brain Tumor Classification \(MRI\)](#). Ce dataset comprend des images IRM des cerveaux de 3 264 individus, incluant ceux avec et sans tumeurs cérébrales. Il contient des cas de meningioma, glioma, tumeurs Pituitary et non-tumeur. Pour illustrer les différents types de tumeurs cérébrales inclus dans notre dataset, voici quelques exemples d'images :

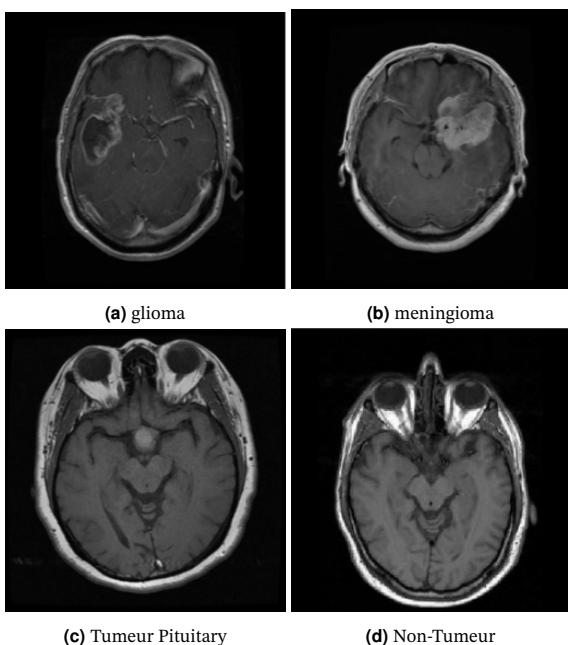


Figure 3. Exemples des différents types de tumeurs cérébrales présents dans le dataset.

Le dataset comprend un total de 3264 images dont 394 images pour le set de test. Le diagramme ci-dessous représente la répartition des images en fonction de leur catégories pour le training set.

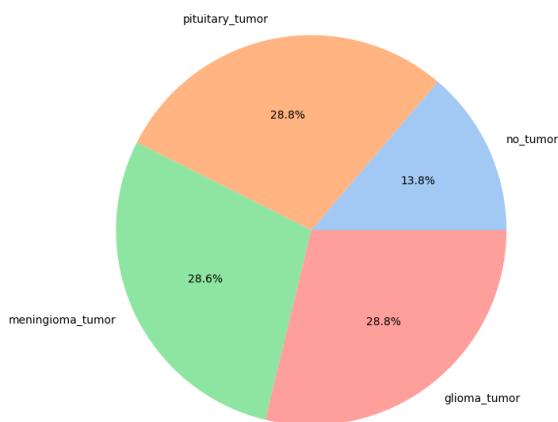


Figure 4. Répartition des types d'images de tumeurs cérébrales dans le dataset de training.

3.1. Prétraitement des données

Le prétraitement des données est une étape cruciale pour assurer la qualité et l'efficacité de l'entraînement des modèles de détection de tumeurs. Cette étape permet de s'assurer que les données sont optimisées pour l'apprentissage des modèles et permet aussi de formater les données. En effet, dans la plupart des cas, les données récupérées ne sont pas identiques: qualité de l'image, taille de l'image, couleurs. Ce sont autant de paramètres qui varient d'une image à une autre et sur lesquels il est possible d'effectuer des traitements pour permettre un apprentissage optimisé pour la machine. Voici les différentes méthodes utilisées et les étapes de pré-traitement réalisées.

3.1.1. Prétraitement des données pour DenseNet et ResNet

Le prétraitement des données pour DenseNet et ResNet ont suivi les étapes suivantes :

- **Redimensionnement des images:** Toutes les images ont été redimensionnées à la taille attendue par DenseNet (224x224 pixels).
- **Normalisation:** Les pixels des images ont été normalisés pour avoir des valeurs comprises entre 0 et 1.

Une fois cela effectué, de l'augmentation des données a été effectué à l'aide de Tensorflow et de Keras. Cela consiste à créer artificiellement des données d'entrées pour aider le modèle à mieux généraliser. Cela s'effectue principalement en choisissant des images dans les données d'apprentissage et les modifier: ajouter du bruit, modifier la taille ou encore tourner l'image.

3.1.2. Prétraitement des données pour EfficientNet

1. **Conversion en niveaux de gris**
L'image couleur est convertie en niveaux de gris pour simplifier le traitement.
2. **Floutage**
Un flou gaussien est appliqué pour réduire le bruit et les petits détails, facilitant ainsi la détection des contours.
3. **Seuillage binaire**
L'image floutée est transformée en une image binaire où les pixels sont soit noirs, soit blancs.
4. **Érosion et Dilatation**
Cette opération réduit les zones blanches pour éliminer les petites taches, puis augmente les zones blanches pour combler les petits trous dans les objets détectés.
5. **Détection des contours**
Les contours des objets blancs sont détectés dans l'image binaire.
6. **Identification du plus grand contour**
Le plus grand contour est sélectionné en fonction de sa surface.
7. **Extraction des points extrêmes**
Les positions des points les plus à gauche, à droite, en haut et en bas du plus grand contour sont déterminées.
8. **Recadrage de l'image**
L'image originale est recadrée selon les positions des points extrêmes du contour pour générer une nouvelle image recadrée.
9. **Traitements des Images d'Entraînement**
Les images d'entraînement de quatre catégories (glioma, meningioma, tumeur pituitaire, et sans tumeur) sont recadrées et sauvegardées dans des répertoires spécifiques.
10. **Traitements des Images de Test**
Les images de test pour les quatre catégories sont également traitées de la même manière pour une évaluation cohérente des modèles de classification d'images.

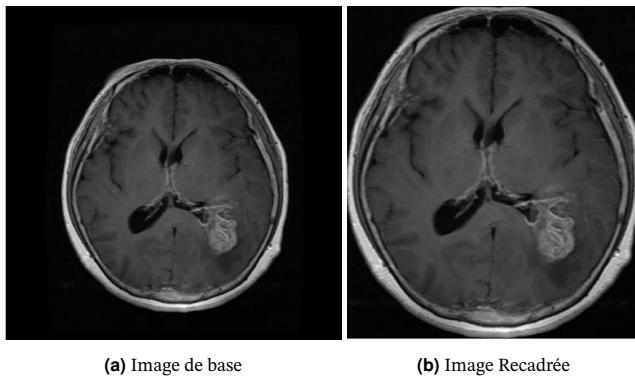


Figure 5. Exemple d'image avant et après le recadrage.

3.2. Interprétation des résultats

3.2.1. Matrices de confusion

Les matrices de confusion sont des outils puissants pour évaluer les performances des modèles de classification. Elles permettent de visualiser non seulement la précision du modèle, mais aussi ses erreurs spécifiques de classification. Voici les composantes d'une matrice de confusion :

- Vrais positifs (TP) : Nombre de fois où le modèle a correctement prédit la classe.
- Faux positifs (FP) : Nombre de fois où le modèle a incorrectement prédit une classe.
- Faux négatifs (FN) : Nombre de fois où le modèle a manqué de prédire la classe correcte.
- Vrais négatifs (TN) : Nombre de fois où le modèle a correctement prédit l'absence de la classe.

		Real Label		Precision = $\frac{\sum TP}{\sum TP + FP}$
		Positive	Negative	
Predicted Label	Positive	True Positive (TP)	False Positive (FP)	Accuracy = $\frac{\sum TP + TN}{\sum TP + FP + FN + TN}$
	Negative	False Negative (FN)	True Negative (TN)	

Recall = $\frac{\sum TP}{\sum TP + FN}$

Figure 6. Precision, recall et accuracy [5]

4. Réseau neuronal convolutif

Avant de définir et tester des modèles, il est important de comprendre l'architecture commune à tous les modèles testés: le réseau neuronal convolutif.

Un réseau neuronal convolutif (CNN) est un type de réseau de neurones qui utilise des opérations de convolution pour extraire automatiquement les caractéristiques d'une image.

Le produit de convolution se définit comme suit:

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x-t)g(t) dt = \int_{-\infty}^{+\infty} f(t)g(x-t) dt \quad (1)$$

Contrairement aux perceptrons multicouches (MLP) où chaque neurone est connecté à tous les neurones de la couche précédente, les neurones des couches convolutionnelles sont connectés uniquement à des régions spécifiques de la couche précédente. L'analyse d'images via des couches entièrement connectées serait impraticable en raison du grand nombre de paramètres requis.

Comparons les paramètres nécessaires pour un MLP et un CNN pour une image de 28x28 pixels en niveaux de gris avec 500 neurones :

- Pour les MLP : $28 \times 28 \times 500 + 500 = 392500$ paramètres
- Pour les CNN : 784 paramètres seulement

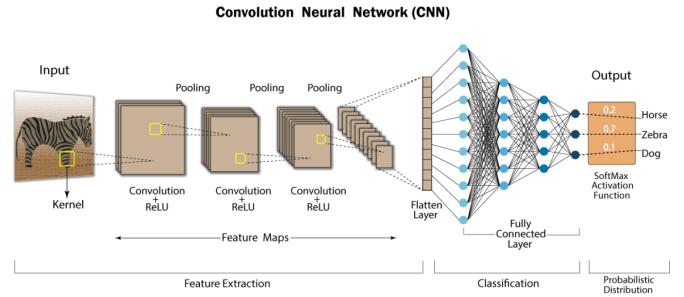


Figure 7. Architecture basique d'un réseau neuronal convolutif [2]

La couche d'entrée reçoit l'image, qui est ensuite traitée par des couches de convolution pour extraire des caractéristiques. Chaque couche de convolution produit une **carte de caractéristiques** (feature map) représentant les informations présentes dans l'image.

Chaque neurone d'une couche de convolution applique le produit de convolution avec un filtre. Les filtres (ou noyaux) sont des petits matrices qui glissent sur les données d'entrée pour appliquer des convolutions. Chaque filtre a des poids spécifiques, et chaque couche de convolution utilise plusieurs filtres pour extraire différentes caractéristiques des données d'entrée.

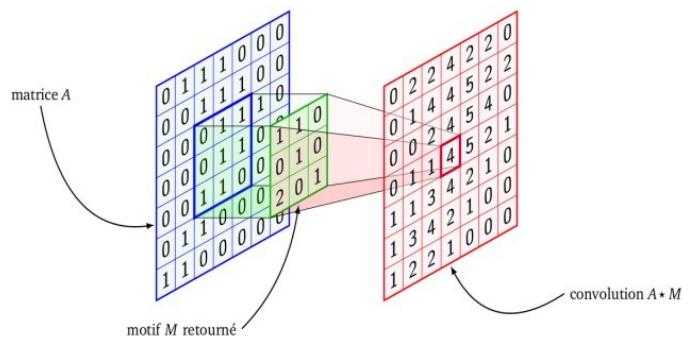


Figure 8. Produit de convolution

Sur cette image, la convolution est appliquée à partir d'un noyau M de taille 3x3 sur une matrice d'entrée A.

Ce type de réseaux utilise généralement plusieurs noyaux par couche de convolution. Chaque noyau apprend à détecter différentes caractéristiques (par exemple, des bords, des textures, des motifs) dans les données d'entrée. Par exemple, si une couche de convolution a 32 noyaux, elle produira 32 cartes de caractéristiques différentes pour la même entrée.

Chaque neurone d'une couche produit donc en sortie une nouvelle matrice ou un tenseur de taille inférieure ou égale à son entrée (cela dépend du padding et stride utilisé) qui résulte du produit de convolution. Avant d'être propagé vers l'avant, cette sortie subit la transformation d'une fonction d'activation.

Une **fonction d'activation** est une fonction mathématique appliquée à la sortie d'un neurone d'un réseau de neurones. Elle

introduit de la non-linéarité dans le modèle, ce qui permet au réseau de capturer des relations non linéaires entre les variables d'entrée et de sortie. Sans fonction d'activation, les couches de neurones successives ne seraient que des transformations linéaires les unes des autres, et le modèle serait incapable d'apprendre des représentations complexes des données.

Dans le cadre d'un réseau de convolution, la fonction d'activation **ReLU** est la plus souvent utilisée:

$$\text{ReLU}(x) = \max(0, x) \quad (2)$$

Une opération de **pooling** est appliquée entre chaque couche: le pooling est une opération qui permet de réduire la dimension des données d'une couche à une autre. Les couches de pooling permettent de conserver les informations essentielles et aide à réduire le nombre de paramètres (donc la durée d'entraînement et la complexité du modèle) et à contrôler le surapprentissage. Une couche de pooling de taille k transforme une entrée de taille $n \times p$ en une sortie de taille $(n/k) \times (p/k)$. Il existe plusieurs types de pooling:

- **MaxPooling:** on ne garde que le **maximum** de chaque sous-matrice de taille $k \times k$
- **MinPooling:** on ne garde que le **minimum** de chaque sous-matrice de taille $k \times k$
- **AvgPooling:** on ne garde que la **moyenne** de chaque sous-matrice de taille $k \times k$

Par exemple, une entrée de taille 100×80 (8000 données) avec un pooling de taille 4 fournit une sortie de taille 25×20 (500 données).

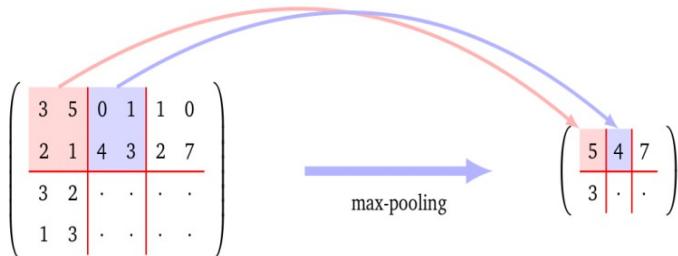


Figure 9. Exemple d'application du max-pooling

Finalement, après avoir été traitées dans plusieurs couches de convolution et de pooling, les cartes de caractéristiques sont aplatis en un vecteur à une dimension. Par exemple, si la dernière carte de caractéristiques a des dimensions $7 \times 7 \times 64$, elle sera aplatie en un vecteur de taille 3136.

Ce vecteur est ensuite injecté dans des couches entièrement connectées qui possèdent une structure similaire aux réseaux MLP. Ces couches combinent toutes les caractéristiques extraites pour effectuer la classification. Dans le cadre d'une classification multi-classes, la fonction d'activation softmax est couramment utilisée afin d'obtenir des probabilités sur les classes prédictes.

La fonction softmax se définit comme suit:

$$\text{Softmax} : f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (3)$$

5. Modèle DenseNet

5.1. Présentation de DenseNet

DenseNet (Densely Connected Convolutional Network) est une architecture de réseau de neurones profonds publiée pour la première

fois en 2017 par Gao Huang, Zhuang Liu, Laurens van der Maaten et Kilian Q. Weinberger.

DenseNet possède la particularité d'utiliser des couches denses entre les différents couches de convolution. En effet, dans DenseNet, chaque couche reçoit en entrée les sorties de toutes les couches précédentes et passe ses propres sorties à toutes les couches suivantes. Dans un réseau convolutif classique, chaque couche prend en entrée uniquement la sortie de la couche précédente.

DenseNet implémente le concept de **concaténation par canal** pour effectuer la propagation des cartes de caractéristiques d'une couche vers une autre. Dans cette architecture, les cartes de caractéristiques sont empilées le long de l'axe des canaux, ce qui signifie que chaque couche produit en sortie un nouveau tenseur avec un nombre plus important de canaux. Ainsi, DenseNet permet une réutilisation maximale des caractéristiques lors de la propagation de caractéristiques, ce qui peut améliorer la performance du modèle.

Cela permet de faciliter la propagation des gradients d'une couche à une autre lors du processus de rétro-propagation. Dans un réseau de neurones profond, il peut y avoir un phénomène de **vanishing gradient**: cela survient lorsque les gradients deviennent tellement petits qu'ils ne peuvent plus être mis à jour. Les connexions directes entre toutes les couches permettent aux gradients de se propager directement vers les couches initiales sans passer par de nombreuses opérations intermédiaires, ce qui réduit le risque d'atténuation des gradients.

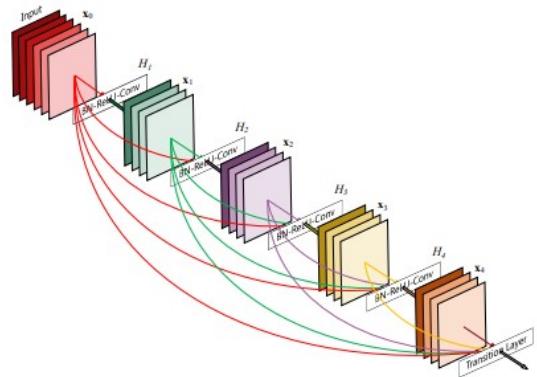


Figure 10. Bloc Dense à 5 couches tel qu'utilisé dans DenseNet [3]

5.2. Avantages

Le principal avantage de ce modèle est la performance liée à l'architecture par blocs. DenseNet possède moins de paramètres que des réseaux classiques du fait de la réutilisation massive des caractéristiques, réduisant ainsi la redondance des informations d'une couche à une autre.

La propagation des gradients est aussi optimisée du fait des connexions directes dans le réseau.

L'architecture DenseNet permet aussi de facilement généraliser, c'est à dire limiter le sur-apprentissage à l'aide de régularisation implicite. En forçant les couches à réutiliser les caractéristiques, DenseNet peut généraliser mieux sur les données de validation et sur des ensembles de données non vues.

5.3. Application au projet

L'implémentation du modèle DenseNet pour ce projet a d'abord été réalisé à l'aide de Tensorflow et de Keras. Keras implémente trois modèles différents nativement:

- DenseNet121

- DenseNet201

Leur différence réside principalement dans le nombre de couches utilisé (le nombre qui est dans le nom du modèle). A noter que vu la profondeur de ces modèles, du sur-apprentissage risque de survenir: le modèle n'arriverait pas à suffisamment généraliser.

Dans un soucis de simplicité, nous laissons Keras se charger de déterminer automatiquement le nombre de labels en lui fournissant le lien vers le dossier qui contient les données. Keras applique ensuite automatiquement un LabelEncoder sur les données catégoriques.

Keras donne aussi la possibilité de charger les modèles avec des poids pré-entraînés sur la base de données ImageNet [6] contenant près de 15 millions d'images représentant plus de 1000 objets différents. Cette base de données est largement utilisée dans le domaine de la vision par ordinateur. L'intérêt d'utiliser des poids pré-entraînés réside sur le fait de pouvoir appliquer le concept d'**apprentissage par transfert** qui consiste à transférer les connaissances apprises lors d'une tâche préliminaire vers une tâche cible. Cela permet au modèle de reconnaître les situations sur lesquelles il a déjà appris et donc de mieux apprendre sur des données similaires permettant ainsi de réduire le temps d'apprentissage puisque le modèle n'a pas à apprendre depuis zéro.

Le modèle est compilé en utilisant une fonction d'optimisation et une fonction de perte:

- **Adam (Adaptive Moment Estimation):** un algorithme d'optimisation utilisé pour entraîner les réseaux de neurones. Il combine les avantages de deux autres extensions de la descente de gradient : AdaGrad et RMSProp.
- **Entropie Croisée:** une fonction de perte qui mesure la dissimilarité entre deux distributions de probabilité : la distribution prédite par le modèle (sortie du modèle) et la distribution réelle (étiquette de classe).

Ces deux fonctions d'optimisation et de perte sont couramment utilisées dans le domaine de l'apprentissage par réseaux de neurones profonds.

5.4. Résultats obtenus

L'apprentissage a été réalisé de trois manières différentes:

- **Apprentissage par Keras:** le pré-traitement des données, le processus d'entraînement et la détermination des hyperparamètres optimaux se fait de manière manuelle.
- **Apprentissage par FastAI:** FastAI est une API de machine learning basé sur PyTorch permettant de facilement déployer et entraîner des modèles de machine learning pré-entraînés sur la base de données ImageNet.
- **Apprentissage manuelle:** L'implémentation de l'architecture du modèle DenseNet121 a été effectué à la main plutôt qu'à l'aide du modèle fourni par Keras puis entraîné avec Keras et Tensorflow.

Nous avons testé ces deux manières afin de comparer les performances d'un modèle pré-entraîné et un modèle entraîné par nos soins.

5.4.1. Entrainement par Keras

L'entraînement s'effectue sur 32 epochs et un batch size de 64. Ce graphique permet de mettre côte à côté l'évolution de la valeur de la fonction de perte et la précision par rapport aux données d'entraînement et aux données de validation, et cela pour chaque epoch.

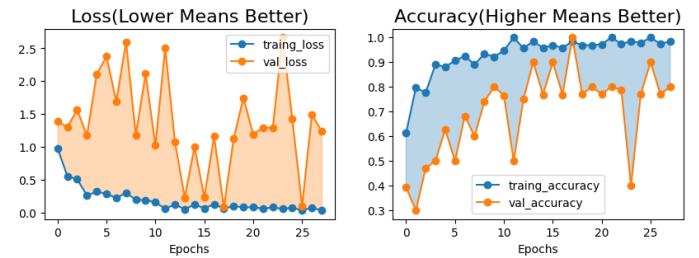


Figure 11. Entrainement du modèle DenseNet avec Keras

L'évolution de la perte pour les données de validation ne diminue pas de manière régulière, ce qui pourrait indiquer un sur-apprentissage: le modèle apprend trop bien les caractéristiques des données de validation et n'arrive pas à généraliser.

Ce comportement est observable aussi lors de l'évolution de la précision par rapport aux données de validation. La précision augmente sur les 10 premiers epochs, ce qui est le comportement attendu, puis se stabilise et varie de manière instable. Cela peut être un signe de sur-apprentissage.

La précision moyenne du modèle est de 0.78, c'est à dire que seulement 78% des images sont correctement classifiées.

Une des solutions pour résoudre le problème de sur-apprentissage est de geler les poids du modèle DenseNet et de n'entraîner que les couches du haut, c'est à dire le réseau fortement connecté.

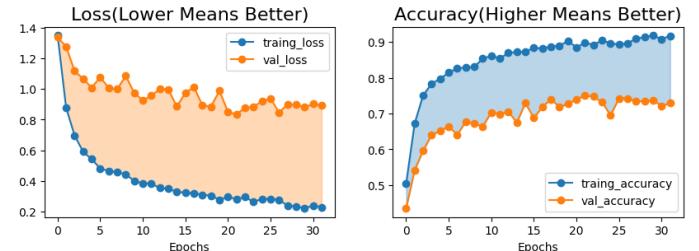


Figure 12. Entrainement du modèle DenseNet avec Keras et paramètres gelés

En effectuant ceci, la précision du modèle descend à seulement 0.68. Contrairement au modèle DenseNet dans lequel tous les poids du modèle étaient ré-entraînés, la précision sur les données d'entraînement augmente de manière régulière mais n'atteint pas 1 aussi rapidement que dans l'autre situation. La précision obtenue par rapport aux données de test augmente dans un premier temps puis se stabilise vers 25 epochs: il n'y a pas de variations brusques comme dans la situation précédente. Cela signifie que le modèle DenseNet, une fois les poids gelés ne souffre plus de sur-apprentissage ou beaucoup moins.

Alertés par le manque de précision du modèle (68%), nous décidons de comparer les résultats obtenus par notre modèle entraîné à la main, avec ceux d'un modèle DenseNet optimisé automatiquement. Pour ce faire, nous utilisons la bibliothèque FastAI.

5.4.2. Entrainement par FastAI

Après avoir ré-entraîné des modèles DenseNet avec des hyperparamètres différents et avec l'aide de FastAI, le modèle DenseNet201 s'est en fait avéré être plutôt efficace pour classifier les différents types de tumeur. Cependant, tous les modèles testés ont de grandes difficultés à prédire la tumeur de type Glioma comme on peut le

remarquer avec cette matrice de confusion:

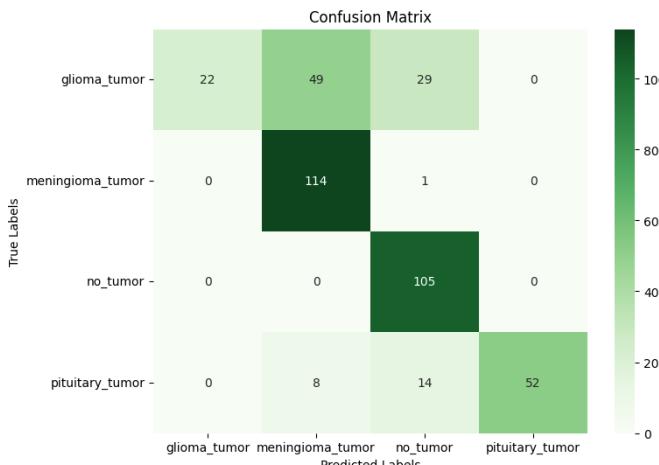


Figure 13. Matrice de confusion du modèle DenseNet201 entraîné via FastAI

De ce fait, la précision moyenne du modèle atteint seulement environ 76% à cause de la difficulté à prédire ce type de tumeur, mais le modèle se montre toutefois très précis quand il s'agit de repérer des tumeurs meningioma et des cerveaux sains, comme le montre cet histogramme:

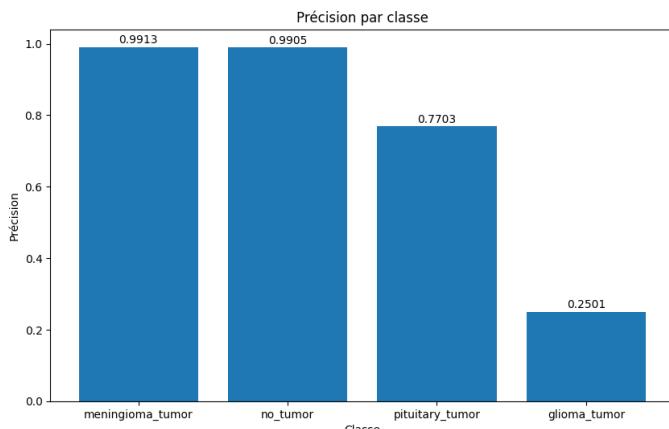


Figure 14. Précision du modèle DenseNet201 en fonction du type de tumeur à classifier

5.5. Implémentation Manuelle de DenseNet-121

En plus d'utiliser des modèles pré-entraînés, nous avons également tenté de coder le modèle DenseNet-121 manuellement en définissant les blocs de convolution et les blocs de transition. Afin de mieux comprendre l'architecture interne de DenseNet. L'objectif était aussi de gagner en flexibilité pour ajuster les paramètres et optimiser les performances du modèle.

Nous avons entraîné ce modèle sur 50 epochs avec un batch size de 32.

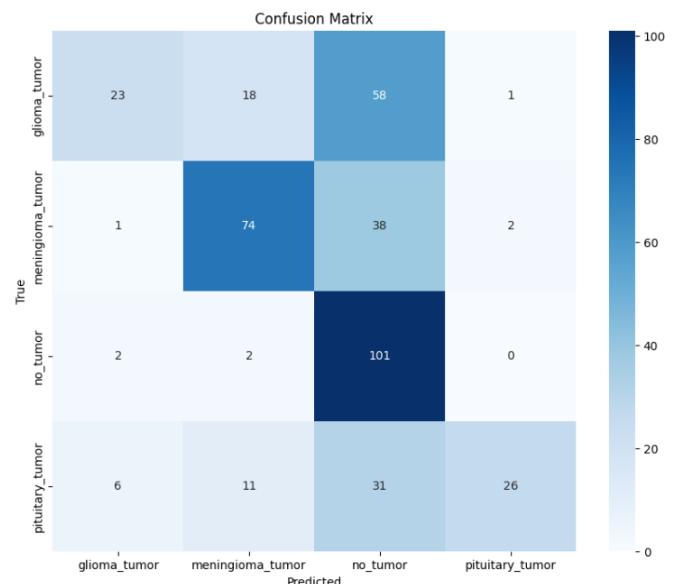


Figure 15. Matrice de confusion du modèle DenseNet121 implémenté à la main

Les résultats sont assez comparables à ceux obtenus avec les deux méthodes précédentes. Cela signifie donc que le problème de précision ne provient pas de nos implémentations respectives, mais simplement du fait que DenseNet n'est pas le modèle le plus approprié pour prédire des tumeurs avec nos données fournies. Nous décidons donc de tester des modèles de type **ResNet**.

6. Modèle ResNet

6.1. Présentation de ResNet

ResNet (ou Residual Network) est une architecture de réseau de neurones convolutifs (CNN) qui a été introduite par Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun dans un article intitulé "Deep Residual Learning for Image Recognition" en 2015. ResNet a par ailleurs remporté le concours ImageNet en 2015 avec une précision record, ce qui a fortement influencé la recherche en vision par ordinateur.

ResNet se distingue par son utilisation de **connexions résiduelles** ou "connexions de saut" (skip connections). Ces connexions permettent de contourner une ou plusieurs couches du réseau et d'ajouter l'entrée directement à la sortie de ces couches, ce qui aide à combattre un phénomène de dégradation des réseaux de neurones profonds appelé "gradient évanescant" (ou vanishing gradient). Ce phénomène est un problème majeur rencontré lors de l'entraînement de réseaux de neurones très profonds, notamment ceux utilisant des fonctions d'activation sigmoïde ou tangente hyperbolique (tanh).

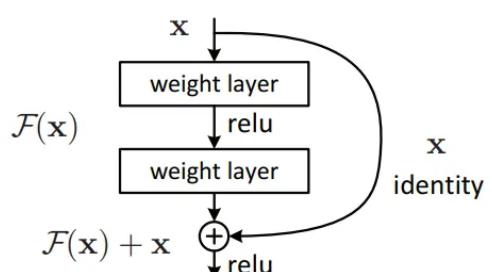


Figure 16. Connexion résiduelle d'un modèle ResNet

Lorsque l'on entraîne un réseau de neurones par rétropropagation, les gradients des erreurs sont calculés et propagés en arrière à travers le réseau pour mettre à jour les poids des neurones. Dans les réseaux de neurones profonds, cette propagation se fait à travers de nombreuses couches. Cependant, à chaque couche, les gradients peuvent diminuer de manière exponentielle en raison des dérivées des fonctions d'activation. Ces dérivées sont comprises entre 0 et 1. Lorsque les gradients sont multipliés successivement par ces petites valeurs à chaque couche, ils peuvent rapidement diminuer vers zéro, surtout dans des réseaux très profonds. Cela entraîne des mises à jour des poids extrêmement petites dans les couches proches de l'entrée du réseau, ralentissant voire arrêtant l'apprentissage.

Les connexions résiduelles du modèle ResNet permettent de limiter ce phénomène en contournant une ou plusieurs couches du réseau et en ajoutant directement l'entrée aux sorties de ces couches pour plusieurs raisons:

- Les connexions résiduelles permettent aux gradients de se propager plus facilement à travers le réseau lors de la rétropropagation. Même si les gradients deviennent très petits dans certaines couches, ils peuvent être ajoutés directement via les connections de saut, ce qui aide à maintenir des gradients de taille significative pour les couches précédentes.
- Les connections résiduelles simplifient l'apprentissage des réseaux profonds en permettant aux couches d'apprendre des résidus ou des différences par rapport à l'entrée initiale, plutôt que d'apprendre une transformation complète de l'entrée. Cela rend l'optimisation plus facile et plus rapide.

La formule de base pour une connexion résiduelle est :

$$y = F(x, \{W_i\}) + x \quad (4)$$

où x est l'entrée, F est la fonction représentant les couches convolutives, et y est la sortie de la connexion résiduelle.

Cette architecture est particulièrement efficace pour de nombreuses raisons :

- **Préservation des informations:** Les connexions résiduelles permettent aux informations initiales de se propager à travers tout le réseau, même dans des couches profondes, réduisant ainsi la perte d'information.
- **Facilité d'optimisation:** En apprenant les résidus (différences entre l'entrée et la sortie souhaitée), les couches intermédiaires peuvent se concentrer sur des modifications plus petites et plus détaillées plutôt que de devoir apprendre des transformations complexes de l'entrée.
- **Évolutivité:** Les réseaux résiduels peuvent être facilement étendus à des profondeurs très importantes (par exemple, ResNet-50, ResNet-101, ResNet-152) tout en maintenant des performances d'apprentissage efficaces.

6.2. Application au projet

ResNet est largement utilisé dans l'imagerie médicale en raison de sa capacité à extraire des caractéristiques complexes à partir d'images médicales, ce qui est crucial pour les tâches de diagnostic.

L'implémentation du modèle ResNet pour ce projet a été réalisée à l'aide de TensorFlow et de Keras. Keras fournit plusieurs variantes de ResNet nativement :

- ResNet50
- ResNet101
- ResNet152

Leur différence réside principalement dans le nombre de couches utilisées (le nombre dans le nom du modèle). Des modèles plus

profonds peuvent capter des caractéristiques plus complexes mais présentent également un risque accru de sur-apprentissage (overfitting), où le modèle n'arrive pas à suffisamment généraliser.

Le modèle est déployé avec la même fonction d'optimisation et de perte que le densenet.

6.3. Résultats obtenus

Nous avons testé plusieurs versions de ResNet incluant les modèles ResNet50, ResNet101 et ResNet152

6.3.1. Entrainement via FastAI

Après avoir testé les 3 modèles et plusieurs hyperparamètres, le modèle ResNet152 s'est avéré être le plus précis dans la classification des tumeurs avec une moyenne de 75,4%. Cependant comme pour DenseNet, la précision du modèle est impacté par la difficulté à classifier les tumeurs Glioma comme le montre la matrice de confusion:

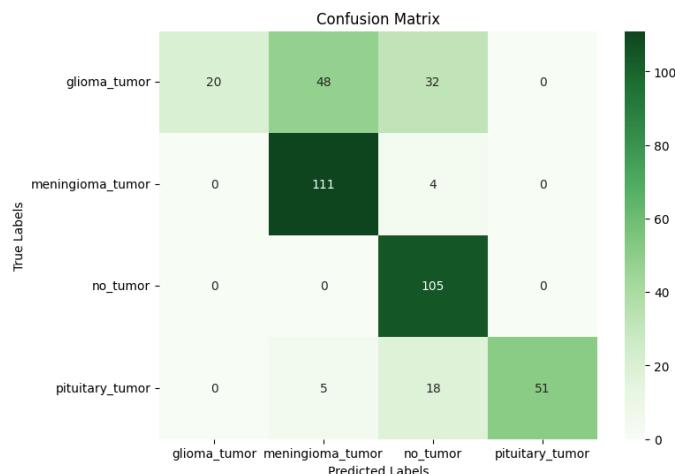


Figure 17. Matrice de confusion pour le modèle ResNet152 entraîné via FastAI

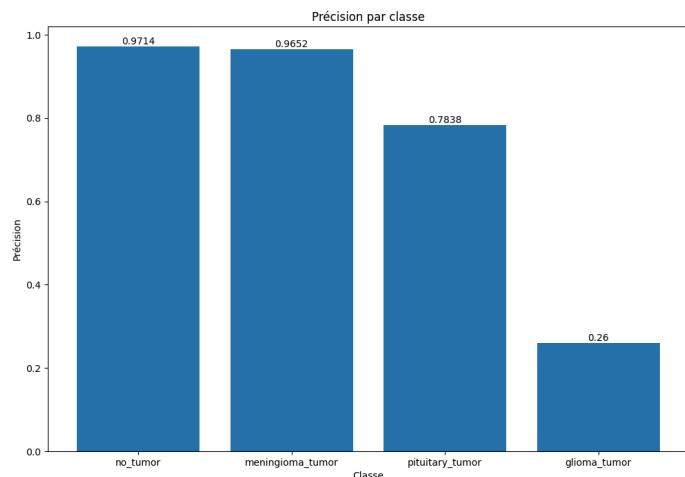


Figure 18. Précision du modèle ResNet152 en fonction du type de tumeur à classifier

6.3.2. Implémentation Manuelle de ResNet

En plus d'utiliser des modèles pré-entraînés, nous avons également tenté de coder le modèle ResNet50 manuellement en définissant les blocs résiduels. Cela nous a permis de mieux comprendre l'architecture interne de ResNet.

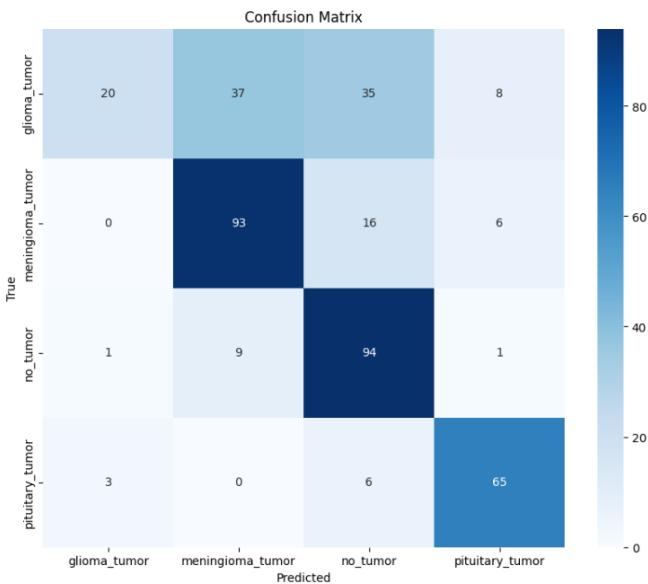


Figure 19. Matrice de confusion pour le modèle ResNet

Bien que l'implémentation manuelle ait permis une flexibilité supplémentaire dans la conception du modèle, les résultats montrent des difficultés similaires dans la classification des tumeurs Glioma.

Bien que ResNet ait montré des performances raisonnables, la complexité et la variabilité des tumeurs cérébrales suggèrent que l'utilisation d'un modèle de détection d'objets pourrait offrir des avantages supplémentaires.

7. Modèle EfficientNet

EfficientNet est une architecture de réseau neuronal convolutif qui optimise les performances en équilibrant la profondeur, la largeur et la résolution du modèle. Développée par Google AI, cette architecture a démontré des performances supérieures en termes de précision tout en nécessitant moins de paramètres et de ressources informatiques comparé à d'autres modèles populaires comme ResNet ou DenseNet.

7.1. Architecture d'EfficientNet

EfficientNet utilise une technique appelée *compound scaling* qui applique un facteur d'échelle uniforme pour augmenter la profondeur (le nombre de couches), la largeur (le nombre de canaux) et la résolution d'entrée des images, de manière équilibrée. Cette approche permet d'obtenir des modèles plus efficaces et performants.

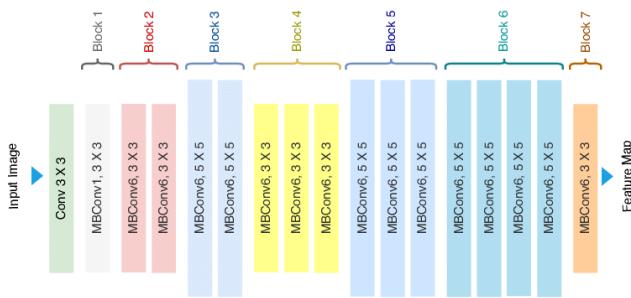


Figure 20. Architecture d'EfficientNet [7]

7.1.1. Compound Scaling

Le *compound scaling* est une méthode systématique pour dimensionner le réseau de manière efficace. Plutôt que de redimensionner uniquement l'une des dimensions (profondeur, largeur ou résolution), EfficientNet redimensionne les trois simultanément selon une règle fixe.

$$\begin{aligned} \text{depth: } d &= \alpha^\phi \\ \text{width: } w &= \beta^\phi \\ \text{resolution: } r &= \gamma^\phi \end{aligned} \quad (5)$$

sous les contraintes :

$$\begin{aligned} \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \quad (6)$$

où ϕ est le facteur d'échelle global, et α, β, γ sont des exposants déterminés empiriquement.

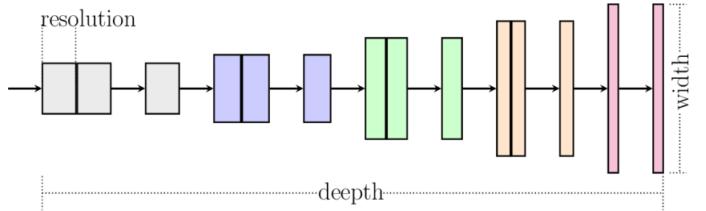


Figure 21. Illustration du Compound Scaling [4]

7.2. Implémentation d'EfficientNetB1

Dans notre projet, nous utilisons EfficientNetB1, une variante de la famille EfficientNet, pour améliorer la détection des tumeurs à partir d'images IRM. Le modèle EfficientNetB1 est pré-entraîné sur ImageNet et modifié pour notre tâche spécifique.

7.2.1. Modification du Modèle

Nous avons apporté les modifications suivantes au modèle pré-entraîné :

- Ajout d'une couche de pooling global (*GlobalAveragePooling2D*).
- Ajout d'une couche de dropout pour réduire le surapprentissage (*Dropout*).
- Ajout d'une couche dense avec une activation softmax pour la classification (*Dense*).

7.2.2. Compilation et Entraînement

Le modèle est compilé avec l'optimiseur Adam, une fonction de perte de type *categorical crossentropy* et des métriques de précision. Nous avons mis en place plusieurs *callbacks* pour améliorer l'apprentissage :

- Un *checkpoint* pour sauvegarder les meilleurs poids du modèle.
- Un *early stopping* pour éviter le surapprentissage.
- Un *scheduler* pour réduire le taux d'apprentissage en cas de stagnation des performances.

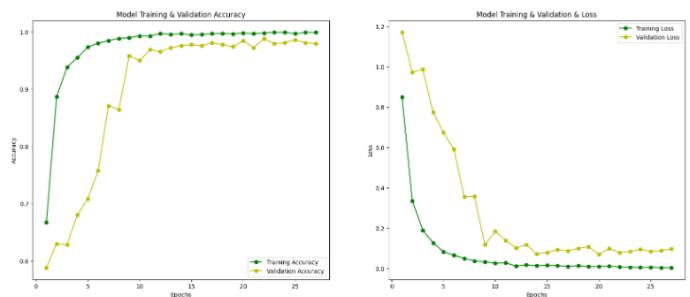


Figure 22. Courbe du training / validation accuracy du modèle

7.2.3. Résultats

Nous avons évalué le modèle sur les données d'entraînement et de test. Les résultats montrent une excellente précision sur les données d'entraînement (1.0) et une bonne précision sur les données de test

(environ 0.815).

Pour mieux comprendre les performances du modèle, nous avons générée une matrice de confusion.

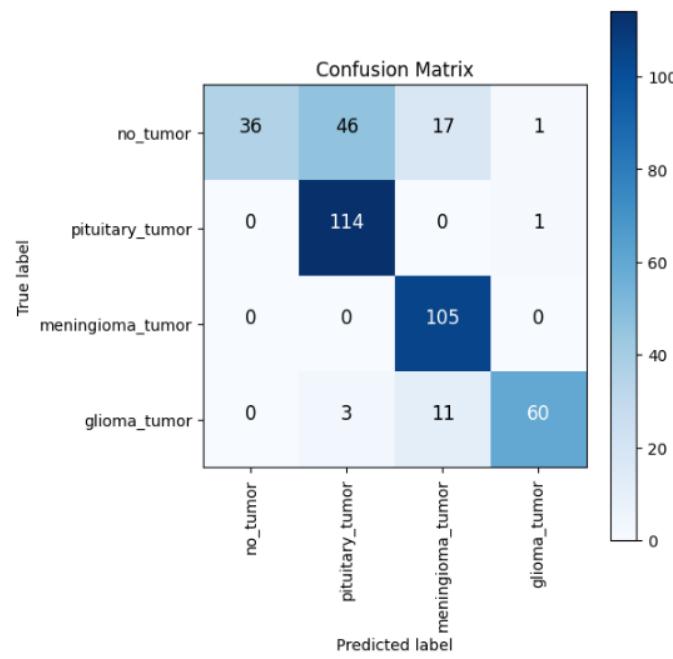


Figure 23. Matrice de Confusion

La matrice de confusion indique que le modèle performe bien sur certaines classes, mais montre des difficultés sur d'autres comme la classe no tumor ou encore glioma, nécessitant potentiellement des ajustements supplémentaires.

7.2.4. Visualisation avec Grad-CAM

Pour interpréter les résultats de notre modèle, nous utilisons Grad-CAM pour visualiser les zones importantes de l'image pour les prédictions du modèle.

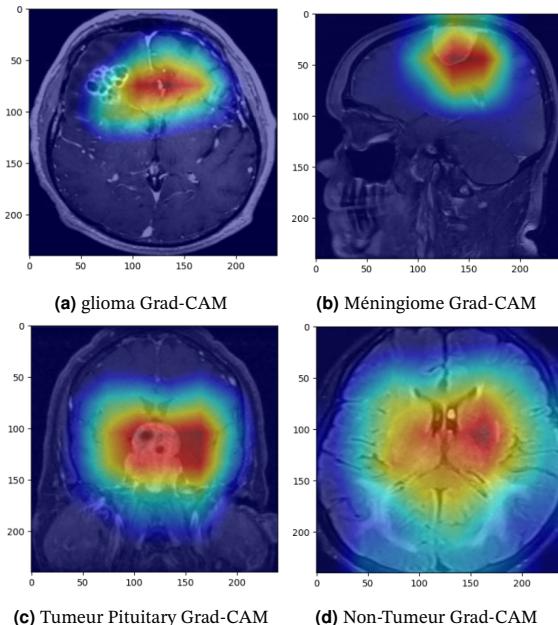


Figure 24. Exemples des différents types de tumeurs cérébrales présents dans le dataset.

Grad-CAM permet de superposer une carte de chaleur sur l'image

d'origine, montrant les régions sur lesquelles le modèle se concentre pour effectuer ses prédictions.

8. Modèle YOLOv8

Le dernier modèle que nous avons réalisé est YOLOv8, un modèle open-source de détection et de segmentation d'images.

8.1. Présentation de YOLO

Le modèle de détection d'images Ultralytics YOLOv8 (You Only Look Once version 8) est une évolution des précédentes versions de YOLO, qui sont des modèles de détection d'objets en temps réel.

L'architecture utilisée par YOLOv8 une architecture de réseau de neurones convolutionnels (CNN) qui prend une image en entrée et la divise en une grille. Chaque cellule de la grille est responsable de la détection des objets dont le centre se trouve dans cette cellule.

Pour chaque cellule de la grille, plusieurs "boîtes d'ancre" sont définies, avec différentes tailles et proportions pour couvrir des objets de différentes dimensions. Chaque boîte d'ancre prédit la probabilité de présence d'un objet et les coordonnées de la boîte englobante.

Pour chaque boîte d'ancre, le modèle prédit :

- Les coordonnées de la boîte englobante (x, y, largeur, hauteur)
- La confiance de la prédiction, c'est-à-dire la probabilité qu'un objet soit présent dans la boîte
- Les classes d'objets possibles avec leur probabilité respective

Après avoir générée toutes les prédictions, une étape de suppression non maximale est appliquée pour éliminer les prédictions redondantes et ne conserver que les boîtes avec la plus haute probabilité.

YOLOv8 est entraîné sur un ensemble de données annotées avec des boîtes englobantes et des classes d'objets. L'objectif de l'entraînement est de minimiser une fonction de perte qui prend en compte la précision des coordonnées des boîtes et la précision de la classification des objets.

YOLOv8 est conçu pour être rapide et précis, ce qui le rend adapté pour les applications en temps réel. Il peut traiter des vidéos et des flux d'images en direct avec une haute efficacité.

8.2. Adapter le dataset

Pour réaliser son entraînement, YOLO nécessite une organisation particulière des données. [8]

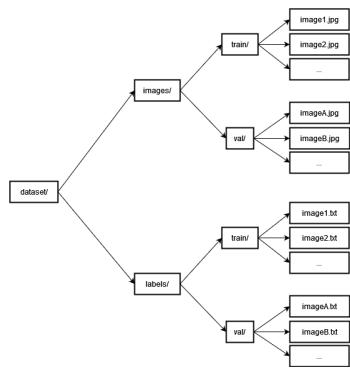
D'abord, le dataset doit être organisé en plusieurs répertoires :

- images/ : qui contiendra toutes les images possédant l'un des formats ['bmp', 'jpg', 'jpeg', 'png', 'tif', 'tiff', 'dng'].
- labels/ : qui contiendra tous les labels associés aux images, au format 'txt'. Chacune d'entre elles possédera un label, avec le même nom (seul le type du fichier change)

On peut encore les découper en 2 sous-dossiers :

- train/ : qui contiendra donc les données d'entraînement
- val/ : qui contiendra les données de validation

Attention, excepté les deux sous-dossiers présentés ici, aucun autre répertoire ne doit être présent, car le modèle ne les parcourra pas : il ne trouvera pas les données.

**Figure 25.** Architecture des répertoires à suivre

Les fichiers textes possédant les labels ont une mise en forme particulière à respecter:

```
1 <class> <x_center> <y_center> <width> <height>
```

Il y a autant de lignes qu'il y a d'éléments à observer sur l'image. De plus, les valeurs numériques sont normalisées. Donc si le centre de l'image est situé à 120x120px pour une image qui fait 240x240px, les valeurs du x et y du centre seront 0.5

Comme présenté précédemment, notre dataset contient des images des tumeurs cérébrales labellisées, mais sans précisions concernant les boîtes englobantes. Deux solutions s'offrent donc à nous :

- Utiliser toute l'image comme boîte englobante : Option la plus facile, elle est aussi de premier abord la moins intéressante. En effet, ne pas avoir de boîtes englobantes adaptées aux tumeurs recherchées va plutôt à l'encontre du fonctionnement de YOLO, et risquerait donc d'introduire des biais importants dans les prédictions.
- Utiliser un des modèles précédemment réalisé pour récupérer des heatmaps des zones sur lesquelles le modèle s'est basé pour prédire la classe, et utiliser ces zones comme boîtes englobantes. Cette possibilité est plus complexe, mais offre une meilleure utilisation des fonctionnalités de YOLO et devrait s'avérer plus performante.

Nous avons finalement choisi de réaliser les deux, pour voir si nos suppositions sont vraies

8.3. Interpréter les résultats

YOLO offre nativement des indications sur l'entraînement qu'il a réalisé. Dans cette analyse, nous allons utiliser 2 résultats

La matrice de confusion, dont le principe de fonctionnement a déjà été présenté plus haut. Attention, cette dernière, fournie par le modèle YOLO lors de son entraînement, inverse le sens des vrais labels et ceux prédits. Comme le code du modèle n'est pas accessible, je ne peux pas les inverser, mais cela a bien été pris en compte pour la lecture des résultats. Pour un souci de logique, les résultats testés après seront présentés par une matrice de confusion qui respecte le même sens (les vrais labels sont sur l'axe x et les labels prédits sur l'axe y).

La courbe Precision-Recall montre la relation entre la précision et le rappel pour différentes classes. C'est une métrique utile pour évaluer les modèles de classification déséquilibrés. Une courbe plus proche du coin supérieur droit indique une meilleure performance. Nous utilisons en particulier la métrique mAP (pour mean Average Precision) car c'est une mesure standard pour évaluer les modèles de détection d'objets, qui prend en compte à la fois la précision et le rappel. Elle fournit une valeur unique qui résume la performance globale du modèle. Elle est une approximation de l'accuracy utilisée

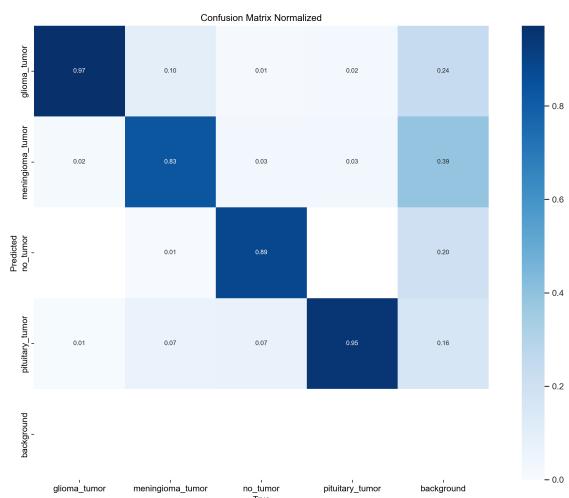
dans les autres modèles dans le contexte de la détection d'objets, car elle reflète la capacité du modèle à détecter et à classer correctement les objets. Notons cependant que mAP est une métrique plus stricte car elle prend en compte les localisations des objets, en plus de l'accuracy d'un modèle qui mesure la justesse de la prédiction de la classe de l'image. Pour avoir un élément visuel pour mieux comprendre la précision et le recall, se référer à la figure suivante : 6

8.4. Premier modèle : utiliser toute l'image comme boîte englobante

Pour ce premier entraînement avec YOLOv8, nous utilisons toute l'image comme boîte englobante. Les valeurs de ces dernières sont donc :

```
1 <class> <x_center> <y_center> <width> <height>
```

8.4.1. Résultats

**Figure 26.** Confusion Matrix Normalized

La matrice de confusion fournie est normalisée, ce qui signifie que les valeurs sont exprimées en pourcentages par rapport au total des prédictions pour chaque classe. Voici une analyse des résultats :

- **Glioma Tumor :**

- Précision : 97% des tumeurs glioma ont été correctement classifiées.
- Faux positifs : 10% des tumeurs meningioma ont été classifiées comme tumeurs glioma, comme 1% des pas de tumeur, et 2% des tumeurs pituitary, soit 23% en tout
- Faux négatifs : 2% des tumeurs glioma ont été classifiées comme tumeurs meningioma et 1% comme des tumeurs pituitary, soit 3% en tout.

- **Meningioma Tumor :**

- Précision : 83% des tumeurs meningioma ont été correctement classifiées.
- Faux positifs : 2% des tumeurs glioma ont été classifiées comme meningioma, comme 3% des pas de tumeur, et 3% des tumeurs pituitary, soit 8% en tout.
- Faux négatifs : 10% des tumeurs meningioma ont été classifiées comme tumeurs glioma, 1% comme pas de tumeur, et 7% comme tumeurs pituitary soit 18% en tout. (Note, il y a 1% de faux positifs de plus que de bonne classification, cela vient des arrondis donnés dans la matrice de confusion)

- **No Tumor :**

- Précision : 89% des cas sans tumeur ont été correctement classifiés.
 - Faux postifs : 1% des tumeurs meningioma ont été classifiés comme pas de tumeur
 - Faux négatifs : 1% ont été classifiés comme tumeurs glioma, 3% comme tumeurs meningioma, et 7% comme tumeurs pituitary, soit 11% des prédictions pour les cas sans tumeur étaient incorrectes.
- Pituitary Tumor :
- Précision : 95% des tumeurs pituitary ont été correctement classifiées.
 - Faux positifs : 1% des tumeurs glioma ont été classifiées comme tumeurs pituitary, comme pour 7% des tumeurs meningioma et 7% des cas sans tumeurs, soit un total de 15% .
 - Faux négatifs : 2% ont été classifiées comme tumeurs glioma, et 3% comme tumeurs meningioma, soit un total de 5% .

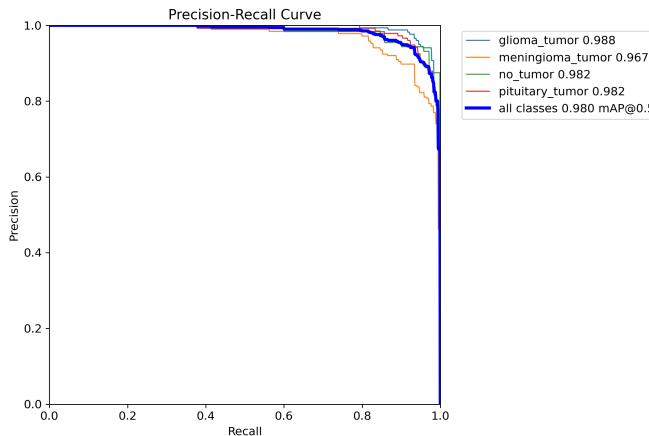


Figure 27. Precision-Recall Curve

Cette courbe montre la relation entre la précision et le rappel (nombre de vrais positifs sur le total des vrais positifs et faux négatifs) :

- Glioma Tumor : Très bonne performance avec une précision de 0.988 et un rappel très élevé.
- Meningioma Tumor : Bonne performance, mais légèrement inférieure à "Glioma Tumor" et "No Tumor".
- No Tumor : Excellente performance avec une précision et un rappel très élevés (0.982).
- Pituitary Tumor : Très similaire à "No Tumor", montrant une haute précision et rappel.
- All Classes : La performance combinée des classes montre un mAP (mean Average Precision) de 0.980 à un seuil de 0.5, indiquant une excellente performance globale.

Ces résultats sont néanmoins à modérer. En effet, ils sont obtenus lors de l'entraînement (d'un dataset bien scindé en train et val de la partie "Training", comme demandé par YOLO), mais lorsque nous réalisons des prédictions avec le set test (fourni par "Testing", et donc complètement inconnu du modèle), les prédictions sont bien moins bonnes.

Nous obtenons ainsi la matrice de confusion ci-dessous

Que l'on peut analyser ainsi :

- Glioma Tumor :
 - Précision : 37% des tumeurs glioma ont été correctement classifiées.

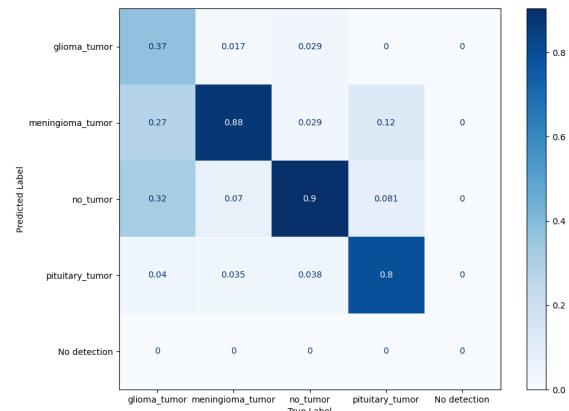


Figure 28. Test confusion matrix normalized

- Faux positifs : 1% des tumeurs meningioma ont été classifiées comme tumeurs glioma, comme 3% des cas sans tumeur, soit 4% en tout
- Faux négatifs : 27% des tumeurs glioma ont été classifiées comme tumeurs meningioma, 32% comme des cas sans tumeurs et 4% comme des tumeurs pituitary, soit 63% en tout.

• Meningioma Tumor :

- Précision : 88% des tumeurs meningioma ont été correctement classifiées.
- Faux positifs : 27% des tumeurs glioma ont été classifiées comme meningioma, comme 3% des pas de tumeur, et 1% des tumeurs pituitary, soit 31% en tout.
- Faux négatifs : 1% des tumeurs meningioma ont été classifiées comme tumeurs glioma, 1% comme pas de tumeur, et 3% comme tumeurs pituitary, soit 5% en tout.

• No Tumor :

- Précision : 90% des cas sans tumeur ont été correctement classifiés.
- Faux postifs : 32% des tumeurs glioma ont été classifiées comme pas de tumeur, comme 7% des tumeurs meningioma et 8% des tumeurs pituitary, soit 47%
- Faux négatifs : 3% ont été classifiés comme tumeurs glioma, 3% comme tumeurs meningioma, et 4% comme tumeurs pituitary, soit 10% des prédictions pour les cas sans tumeur étaient incorrectes.

• Pituitary Tumor :

- Précision : 80% des tumeurs pituitary ont été correctement classifiées.
- Faux positifs : 4% des tumeurs glioma ont été classifiées comme tumeurs pituitary, comme pour 3% des tumeurs meningioma et 4% des cas sans tumeurs, soit un total de 11% .
- Faux négatifs : 12% a été classifiée comme tumeurs meningioma, et 8% comme sans tumeur, soit un total de 20%

Les résultats en conditions de tests sont bien plus proches des résultats obtenus avec les autres modèles, et restent tout de même corrects pour la plupart des types de cancers, avec une accuracy générale de 74,1%. Seules les tumeurs glioma sont très difficiles à prédire, et le modèle YOLO fournit ici parmi les meilleurs taux de réussite.

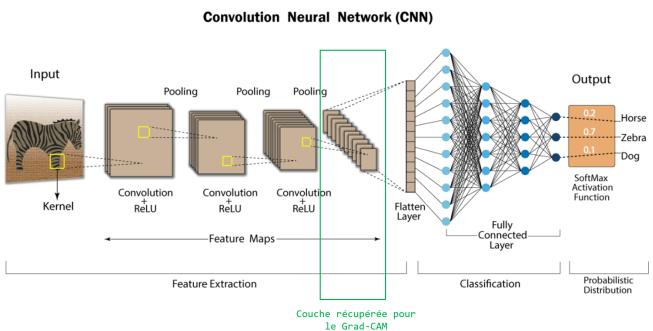


Figure 29. Architecture d'un modèle CNN et couche récupérée pour Grad-CAM [2]

8.5. Second modèle : utiliser des heatmaps des zones intéressantes pour avoir les boîtes englobantes

8.5.1. Récupération des zones intéressantes

Cette fois, nous essayons d'obtenir des boîtes englobantes des zones de tumeurs des images. Pour cela, nous utilisons Grad-CAM (Gradient-weighted Class Activation Mapping).[1] Cette technique est utilisée pour visualiser et comprendre les décisions prises par les modèles de réseaux de neurones convolutionnels (CNN). Elle aide à identifier quelles parties d'une image influencent le plus les prédictions du modèle. Fonctionnement de Grad-CAM :

1. Identification de la couche de convolution : Sélectionner une couche de convolution intermédiaire du réseau, généralement la dernière couche de convolution, car elle capture des caractéristiques de haut niveau. On récupère donc la dernière couche avant les couches entièrement connectées (fully connected layers), comme présenté sur la figure 29
2. Calcul des gradients : Calculer les gradients de la classe d'intérêt par rapport aux caractéristiques de cette couche de convolution. Ces gradients indiquent l'importance des caractéristiques pour la classe d'intérêt.
3. Pondération des caractéristiques : Moyennant ces gradients, on obtient une carte de pondération pour chaque canal des caractéristiques de la couche sélectionnée. Cela permet de mesurer l'importance de chaque canal pour la décision finale.
4. Génération de la carte de chaleur (heatmap) : Combiner les cartes pondérées des caractéristiques pour obtenir une carte de chaleur 2D. Cette carte de chaleur montre les régions de l'image d'entrée qui ont le plus contribué à la prédiction.
5. Superposition sur l'image d'origine : La carte de chaleur est ensuite redimensionnée pour correspondre à la taille de l'image d'entrée et superposée à l'image originale, mettant en évidence les régions importantes.

Pour récupérer ces cartes de chaleurs, nous utilisons l'API "core" de "tf-explain", et plus particulièrement la librairie "GradCAM".[9] Notons que nous n'utilisons pas le modèle YOLO réalisé juste avant car c'est un modèle pour la détection d'objets qui prédit des boîtes englobantes et des classes d'objets directement à partir des images en une seule passe. Sa structure et son mode de fonctionnement sont donc très différents de ceux des modèles de classification d'images typiques. GradCAM est conçu pour fonctionner avec des couches convolutionnelles qui sont couramment utilisées pour la classification. Il s'appuie sur les gradients de la sortie par rapport aux activations d'une couche spécifique pour produire une heatmap. YOLO, bien qu'il utilise des couches convolutionnelles, n'est pas principalement un modèle de classification, et ses sorties sont structurées différemment (boîtes englobantes, confiances, et classifications). Nous utilisons donc comme modèle de référence EfficientNet, présenté juste avant. Sa structure est plus simple, car il ne prédit que les classes, et c'est avec ce modèle que nous obtenons les meilleurs résultats, et donc nous devrions avoir les meilleures heatmaps.

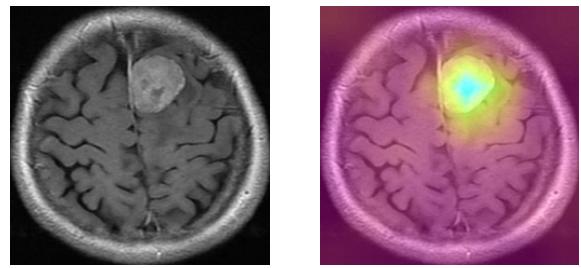


Figure 30. Exemple d'une heatmap exploitable

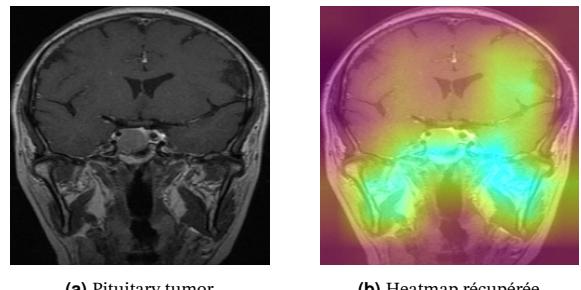


Figure 31. Exemple d'une heatmap moyennement fiable

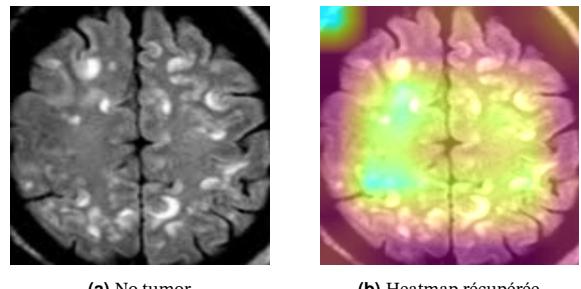


Figure 32. Exemple d'une heatmap obtenue pour une image ne présentant pas de tumeur

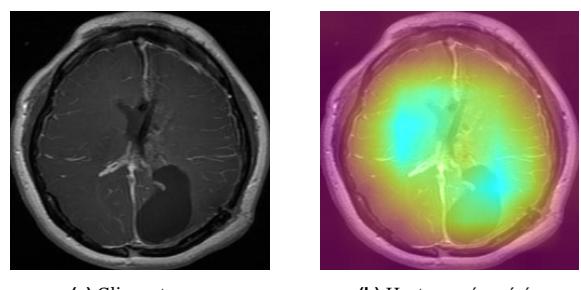


Figure 33. Exemple d'une heatmap inexploitable

On observe immédiatement certains problèmes. Comme on récupère une couche d'un modèle, on récupère également ses biais et erreurs. Bien que les labels des classes soient tous corrects (car récupérés dans le nom des répertoires dont proviennent les images), les cartes de chaleurs peuvent être fausses. En effet, le modèle peut se concentrer sur les mauvaises zones de l'image, principalement concernant les classes pour lesquelles le modèle n'est pas très fiable. Les cartes de chaleur de la classe des Glioma tumor, les plus difficiles à prédire, sont donc généralement moins exploitables. On remarque également que dans le cas des images ne présentant pas

de tumeur, les heatmap sont plus grosses sur l'image, puisqu'il n'y a pas de zone en particulier à montrer. On peut donc craindre que les images n'ayant pas obtenu des heatmaps assez centrées sur la tumeur soient confondues avec celles ne possédant pas de tumeur.

8.5.2. Création des boîtes englobantes

Une fois les cartes de chaleur récupérées, il faut à présent créer des boîtes englobantes à partir des données obtenues. Pour réaliser cela, on commence par mettre nos heatmap en noir et blanc, pour favoriser les contrastes, et pouvoir appliquer la fonction "cv2.threshold" et "cv2.findContours". La première fonction permet de réaliser un masque binaire : Tous les pixels dont l'intensité est supérieure ou égale à la valeur "threshold" choisié sont définis à 255 (blanc), et tous les autres pixels sont définis à 0 (noir). Ensuite, findContours est utilisé pour trouver les contours dans le masque binaire. Les contours détectés sont des listes de points qui délimitent les régions blanches (255) de l'image binaire. On ne garde que les points d'extrémité des contours externes, dont on pourra ensuite réaliser des boîtes englobantes carrées. La valeur du threshold, qui détermine donc quelle intensité de la heatmap sera gardée ou non et par extension quelles zones sont gardées, est une valeur sensible, car elle peut varier beaucoup d'une image à l'autre.

D'abord, il a fallu limiter le "bruit" qui provenait de la nature des images IRM, avec les bords et les os naturellement clairs. Pour réaliser ceci, nous nous sommes basés sur la taille et la proportion des boîtes trouvées. Si elles sont trop petites ou trop disproportionnées, on ne les garde pas. Augmenter la valeur de bascule n'était ici pas une bonne idée, car on perdait vite les boîtes intéressantes.

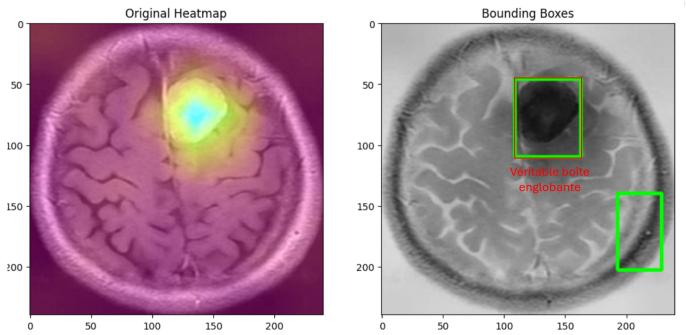


Figure 35. Threshold trop petit

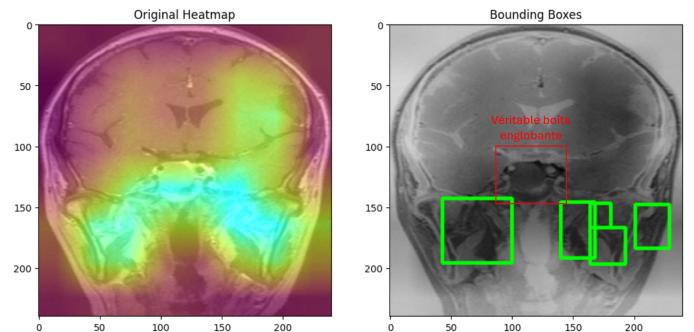


Figure 36. Threshold trop grand

Lorsque nous ne trouvions aucune boîte englobante avec le threshold choisi pour la classe, toute l'image est prise comme boîte (comme pour le premier modèle).

8.5.3. Résultats

Plusieurs essais ont été réalisés.

D'abord en ne gardant qu'une boîte englobante (la plus grande et la plus proportionnellement carrée, puisque les boîtes nuisances sont généralement assez petites et rectangulaires)

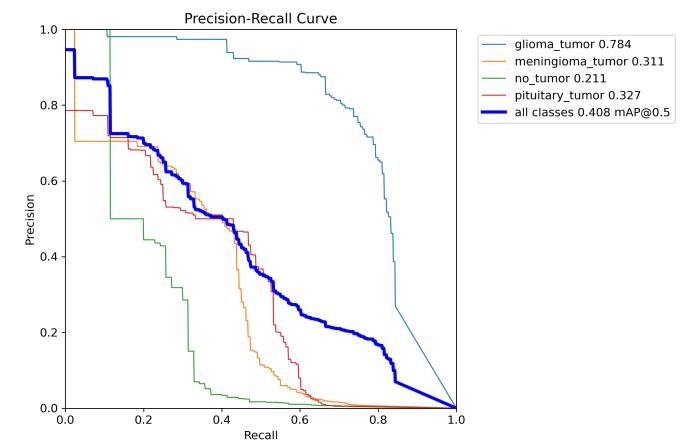


Figure 37. Premier essai avec des boîtes englobantes précisées

Ensuite, un test en gardant plus de boîtes englobantes a été réalisé, pour être sûre d'avoir la bonne, mais en induisant un biais avec les boîtes qui englobent des parties saines du cerveau.

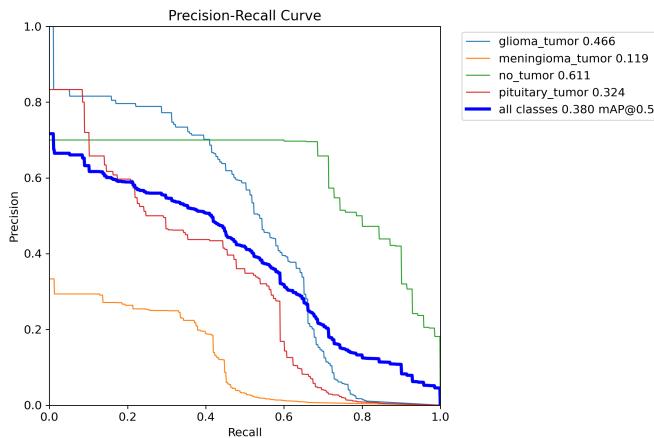


Figure 38. Premier essai avec des boîtes englobantes précisées

Dans le deux cas, les résultats sont extrêmement faibles, avec une précision qui ne dépasse même pas 50%.

De nombreux éléments, mis en avant tout au long de l'explication de la démarche, viennent expliquer des résultats aussi navrants :

- Les cartes de chaleur qui sont, dès le début, pas toujours bien centrées sur les zones intéressantes de l'image (ici les tumeurs).
- Les boîtes englobantes, qui sont déduites avec une valeur de threshold pas toujours adaptée

Pour y remédier, de nombreuses options existent, comme adapter le threshold à la forme de l'image. On récupérerait donc d'abord le bord de l'IRM, et on pourrait réaliser des clusters des différents points de vue. Récupérer les bords de l'IRM se ferait très facilement avec les images initiales, qui ont toutes un bord blanc net qui sépare le cerveau du bord noir de l'image. Ici, pas besoin d'avoir des labels de ces différents points de vue, cette classification n'est utile que pour déterminer la valeur du point de bascule. On réalisera ensuite des essais de threshold par classe de tumeur et par prise de vue. Pour déterminer la meilleure valeur, on pourrait commencer par calculer la position de la boîte englobante : pour une même prise de vue, les tumeurs sont sensiblement au même niveau de l'image. Par exemple, les tumeurs pituitaire sont très centrées dans le cerveau. Pour aller encore plus loin, on pourrait commencer par apprendre à notre modèle à reconnaître le cerveau dans l'IRM, il ne nous indiquerait donc plus les cervicales lorsque la vue est prise de dos par exemple.

Ces adaptations permettraient déjà d'avoir un set de labels bien plus corrects.

9. Script python bonus

9.1. Description de l'Application

L'application de classification d'images est conçue pour permettre à l'utilisateur de sélectionner une image et de la classer à l'aide de plusieurs modèles de réseaux de neurones pré-entraînés. L'application est développée en Python en utilisant les bibliothèques Tkinter pour l'interface utilisateur, OpenCV pour le traitement d'image, TensorFlow pour le chargement des modèles Keras et PyTorch pour le modèle YOLO.

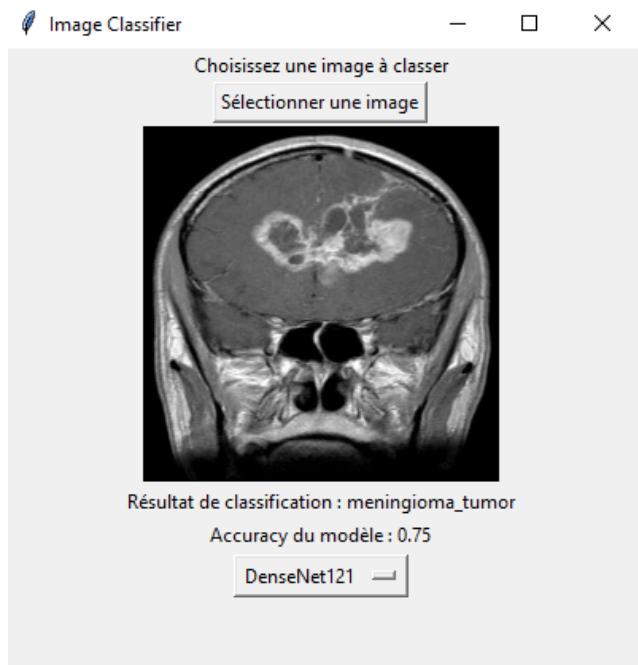
9.2. Fonctionnalités de l'Application

- **Sélection de l'image :** L'utilisateur peut sélectionner une image à partir de son système de fichiers.
- **Affichage de l'image :** L'image sélectionnée est affichée dans l'interface utilisateur.
- **Chargement du modèle :** L'utilisateur peut choisir parmi plusieurs modèles de classification, notamment DenseNet121, ResNet50, EfficientNetB0 et YOLO.

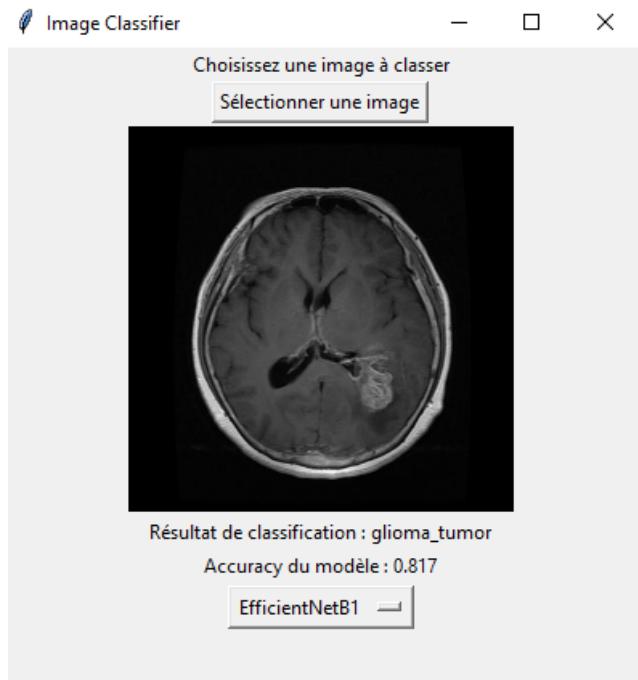
- **Prédiction de l'image :** Une fois l'image sélectionnée, elle est redimensionnée et traitée selon les spécifications du modèle choisi, puis classée.
- **Affichage des résultats :** Les résultats de la classification ou de la détection sont affichés dans l'interface utilisateur.
- **Accuracy du modèle :** L'accuracy du modèle sélectionné est affichée.

9.3. Résultats

Les figures ci-dessous montrent des exemples d'images avant et après la classification ou la détection, ainsi que l'interface utilisateur de l'application.



(a) Résultat avec DenseNet121



(b) Résultat avec EfficientNetB1

Figure 39. Interface utilisateur et exemple de résultat de classification.

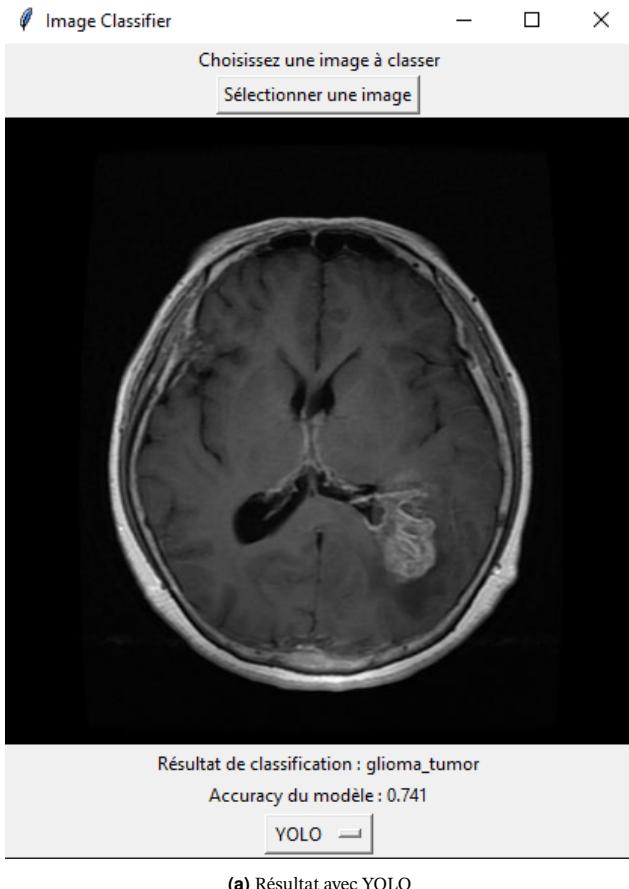


Figure 40. Exemple d'image avec YOLO.

10. Améliorations possibles et futures

Bien que notre projet ait démontré des résultats prometteurs, il existe plusieurs axes d'amélioration qui pourraient être explorés pour améliorer la précision et la robustesse des modèles utilisés pour la classification des tumeurs cérébrales.

10.1. Augmentation des données

L'augmentation des données est une technique essentielle pour améliorer la performance des modèles de deep learning, surtout lorsqu'on travaille avec des jeux de données limités. Pour aller plus loin, nous pourrions effectuer:

- **Augmentation avancée :** Utiliser des techniques d'augmentation plus sophistiquées telles que la distortion géométrique, la modification des couleurs, et l'application de filtres de flou ou de bruit.
- **Synthèse d'images :** Générer des images synthétiques de tumeurs cérébrales en utilisant des techniques comme les réseaux antagonistes génératifs (GANs) pour augmenter la diversité des données d'entraînement.

10.2. Enrichissement du dataset

Utiliser des bases de données plus grandes et plus diversifiées pourrait améliorer la capacité de généralisation des modèles. Par exemple, intégrer des jeux de données provenant de différentes institutions médicales pour avoir une représentation plus variée des types de tumeurs et des techniques d'imagerie.

10.3. Optimisation des modèles

Explorer et tester d'autres architectures de réseaux de neurones pourrait mener à de meilleures performances :

- **Modèles hybrides :** Combiner les points forts de plusieurs architectures, par exemple en utilisant des caractéristiques extraites d'un modèle comme entrée pour un autre modèle.
- **Recherche d'architecture neuronale (NAS) :** Utiliser des algorithmes automatiques pour rechercher les architectures optimales pour notre tâche spécifique.
- **Ensembles de modèles (ensemble learning) :** Utiliser des techniques d'ensemble pour combiner les prédictions de plusieurs modèles afin d'améliorer la robustesse et la précision globale.

10.4. Amélioration des techniques de prétraitement

Le prétraitement des données joue un rôle crucial dans la performance des modèles. Voici quelques améliorations possibles :

- **Segmentation des tumeurs :** Implémenter des techniques de segmentation pour isoler les tumeurs des images IRM avant la classification, ce qui pourrait réduire le bruit et améliorer la précision du modèle.
- **Normalisation avancée :** Utiliser des techniques de normalisation plus sophistiquées adaptées aux caractéristiques spécifiques des images médicales.

10.5. Optimisation des hyperparamètres

Une optimisation plus approfondie des hyperparamètres pourrait également améliorer les performances des modèles :

- **Recherche bayésienne :** Utiliser des méthodes d'optimisation plus avancées comme la recherche bayésienne pour trouver les hyperparamètres optimaux.
- **Validation croisée :** Mettre en œuvre une validation croisée plus rigoureuse pour mieux évaluer les performances des modèles et éviter le surapprentissage.

10.6. Utilisation de modèles de détection d'objets

Comme mentionné précédemment, des modèles de détection d'objets comme YOLO peuvent être utilisés pour détecter et classifier simultanément les tumeurs. L'optimisation de ces modèles pour notre cas d'utilisation spécifique pourrait apporter des améliorations significatives.

- **Fine-tuning de YOLO :** Ajuster les hyperparamètres et entraîner le modèle YOLO sur un ensemble de données plus large et mieux annoté.
- **Utilisation de boîtes englobantes plus précises :** Améliorer la précision des boîtes englobantes utilisées pour l'entraînement en exploitant les cartes de chaleur générées par des modèles de classification.

10.7. Collaboration interdisciplinaire

Pour améliorer nos modèles de prédiction, collaborer avec des neurologues et des radiologues est essentiel :

Compréhension clinique :

- Réunions avec des experts pour comprendre leurs techniques de détection.
- Analyse des cas réels pour identifier les caractéristiques importantes.

Intégration médicale :

- Collaborer pour obtenir des annotations précises.
- Adapter les modèles aux critères diagnostiques cliniques.

Validation et feedback :

- Impliquer des experts médicaux dans la validation des modèles.
- Mettre en place un système de feedback continu.

Formation mutuelle :

- Éduquer mutuellement les chercheurs et les médecins sur leurs domaines respectifs.
- **Interface utilisateur intuitive** : Une interface conviviale pour faciliter l'utilisation par les professionnels de santé.
- **Rapports automatisés** : Génération automatique de rapports de diagnostic basés sur les prédictions des modèles.
- **Intégration avec les systèmes de santé** : Connecter l'application aux systèmes de gestion des informations de santé (HIS) pour un flux de travail intégré et efficace.

En poursuivant ces pistes d'amélioration, nous pourrions non seulement augmenter la précision et la robustesse de nos modèles, mais aussi contribuer de manière significative à l'amélioration des techniques de diagnostic des tumeurs cérébrales, offrant ainsi un soutien précieux aux professionnels de santé et aux patients.

10.8. Développement d'une application clinique

Développer une application clinique intégrant les modèles de classification pour assister les radiologues dans le diagnostic des tumeurs cérébrales.

11. Conclusion

Le diagnostic de tumeurs cérébrales à l'aide de techniques de machine learning et de deep learning représente une avancée significative dans le domaine de la santé. Ce projet a démontré l'efficacité de diverses architectures de réseaux de neurones, telles que ResNet, EfficientNet et YOLO, pour la classification des tumeurs à partir d'images IRM. En explorant les avantages et les défis de chaque modèle, nous avons pu identifier les points forts et les limitations de nos approches actuelles.

Voici les résultats obtenus pour chacun des modèles:

Modèle	Accuracy sur le test set
Densenet121	75%
Densenet201	76%
ResNet50	75.4%
ResNet101	74.9%
ResNet152	75.4%
EfficientNetB1	81.7%
YOLO	74,1%
YOLO avec heatmaps	40,8%
Moyenne	76,1%

Table 1. Résultats obtenus pour chaque modèle

Bien que les résultats obtenus soient prometteurs, des améliorations supplémentaires sont nécessaires pour augmenter la précision et la robustesse des modèles. En particulier, l'augmentation et l'enrichissement des données, l'optimisation des hyperparamètres, et l'exploration de nouvelles architectures de réseaux de neurones pourraient contribuer à des performances encore meilleures.

Enfin, la collaboration interdisciplinaire avec des experts en radiologie et en oncologie sera essentielle pour affiner nos modèles et assurer leur pertinence clinique. En développant une application clinique intuitive et intégrée, nous pouvons faciliter l'adoption de ces technologies par les professionnels de santé et améliorer les soins aux patients.

En conclusion, ce projet représente une étape importante vers l'utilisation de l'intelligence artificielle pour le diagnostic des tumeurs cérébrales. Avec des efforts continus et des améliorations, nous espérons contribuer à des diagnostics plus précis et rapides, offrant ainsi un soutien précieux aux professionnels de santé et améliorant les résultats pour les patients.

■ References

- [1] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Oct. 2019, ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. [Online]. Available: <http://dx.doi.org/10.1007/s11263-019-01228-7>.
- [2] S. Bhardwaj, *Convolutional Neural Networks : Understand the Basics*. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-understand-the-basics/>.
- [3] *Densely Connected Convolutional Networks*. [Online]. Available: <https://arxiv.org/pdf/1608.06993.pdf>.
- [4] *Efficient net compound scaling on three parameters*. [Online]. Available: https://www.researchgate.net/figure/Efficient-net-compound-scaling-on-three-parameters-Adapted-from-Tan-and-Le-2019_fig3_342580295.
- [5] *Evaluating ML Models: The Confusion Matrix, Accuracy, Precision, Recall*. [Online]. Available: <https://medium.com/deeplearn/evaluating-ml-models-the-confusion-matrix-accuracy-precision-recall-b3f8f4431fa5>.
- [6] *ImageNet*. [Online]. Available: <https://www.image-net.org/>.
- [7] K. Maurya, *The architecture of Efficient*. [Online]. Available: https://www.researchgate.net/figure/The-architecture-of-Efficient_fig2_370729188.
- [8] *Object detection datasets overview*. [Online]. Available: <https://docs.ultralytics.com/datasets/detect/>.
- [9] *Tf-explain usage*. [Online]. Available: <https://tf-explain.readthedocs.io/en/latest/usage.html>.