



M2 Data Science
Apprentissage Statistique

Rapport de Projet
Classification de tumeurs cérébrales via CNNs

Lilya-Nada KHELID, Sarah OUAHAB,
Erivan INAN, Sheïma MEBARKA

Professeurs : Erwan SCORNET, Ismaël CASTILLO

Un rapport soumis en vue de l'accomplissement des exigences de
l'ISUP et Sorbonne Université pour l'obtention du diplôme de
Master 2 Data Science

Abstract

Les tumeurs cérébrales représentent un enjeu majeur de santé publique, nécessitant des outils de diagnostic précis et rapides pour améliorer la prise en charge des patients. Ce projet vise à développer un réseau neuronal convolutionnel (CNN) sur mesure et à réaliser un benchmark avec des modèles pré-entraînés tels qu'ImageNet et InceptionV3 pour la classification automatisée des tumeurs cérébrales à partir d'images IRM. Ce projet met en lumière le potentiel de l'intelligence artificielle pour assister les professionnels de santé dans la détection précoce et l'évaluation des pathologies cérébrales.

L'ensemble des fichiers codes développés pour ce projet est consultable [sur cette page Github](#).

Keywords : Classification, Convolutional Neural Network, Deep Learning, Explicability, Medical Imaging, Magnetic Resonance Imaging (MRI), Tumor, Benchmarking

Table des matières

1	Introduction	1
2	Revue de l'État de l'Art	3
3	Présentation du jeu de données	5
4	Data Visualisation	8
4.1	Preprocessing	8
4.2	Équilibre des classes	9
4.3	Représentation graphique des données	9
4.4	Analyse de la variance	10
4.4.1	Variance intra-classes	10
4.4.2	Variance inter-classes	10
4.4.3	Ratio inter/intra-classes	10
4.5	Architecture et Arborescence du projet	11
5	Métriques utilisées pour le benchmarking	12
5.1	Définition des Métriques	12
5.2	Métriques Globales	13
6	Modèles	14
6.1	Baseline model : Inception V3	14
6.2	Modèle ReLU	18
6.3	Modèle Augmented ReLU	20
6.4	Modèle Sigmoïd	21
6.4.1	Implémentation du modèle	21
7	Réultats et Benchmark	23
7.1	ReLU vs Augmented ReLU	23
7.2	ReLU vs Sigmoid	24
7.3	Deux types de lecture	25
7.3.1	Optimisation de la Performance Globale de Classification	25
7.3.2	Minimisation des Risques Cliniques	25
8	Explicabilité et Interprétabilité des modèles	27
9	Conclusion	31
10	Discussion	32
10.1	Limitations	32
10.2	Axes d'amélioration	33
Appendices		35

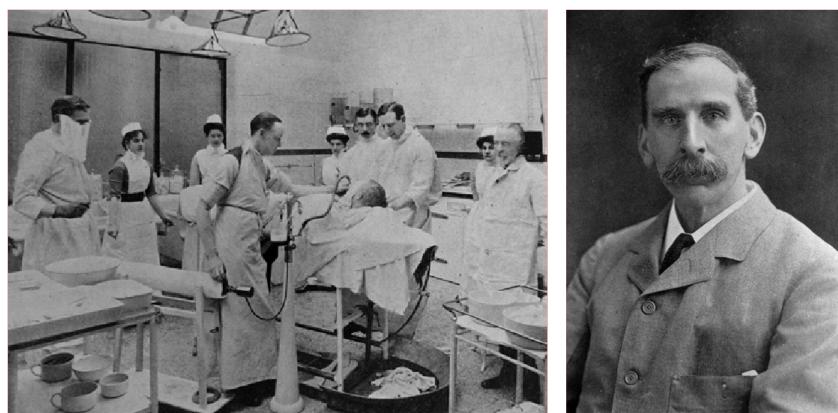
Chapitre 1

Introduction

Contexte historique

Les tumeurs cérébrales constituent un enjeu de santé publique majeur, avec des implications importantes pour le diagnostic et le traitement des patients. Les méthodes traditionnelles de diagnostic, bien que robustes, reposent fortement sur l'expertise des radiologues et des cliniciens, ce qui peut conduire à des retards ou à des erreurs dans l'identification des tumeurs. Ces défis sont exacerbés par l'augmentation du volume de données d'imagerie médicale, notamment les images IRM, disponibles pour chaque patient.

L'histoire du diagnostic et du traitement des tumeurs cérébrales remonte à plusieurs siècles, avec des avancées marquantes dans la compréhension et la prise en charge de ces affections. Au début du XIX^e siècle, elles étaient principalement identifiées lors d'autopsies, les techniques d'imagerie n'existant pas encore. Les premières tentatives chirurgicales pour retirer des tumeurs cérébrales ont été réalisées dans les années 1880 grâce aux travaux pionniers de Sir Victor Horsley, un neurochirurgien britannique. Ces interventions étaient souvent associées à des taux de mortalité élevés en raison de l'absence de techniques d'anesthésie et de stérilisation avancées.



Sir Victor Horsley, juste avant de commencer une intervention chirurgicale dans le bloc opératoire du Queen Square Hospital, à Londres, en 1906. Debout, sur le bord droit de la photographie, le professeur Theodor Kocher est visible en train d'assister à l'intervention.
[Horsley (1906)]

Avec l'émergence de l'intelligence artificielle au XXI^e siècle, de nouvelles opportunités se sont ouvertes pour analyser les images IRM à grande échelle. Les algorithmes d'apprentissage automatoque, et plus particulièrement les réseaux neuronaux convolutionnels (CNN) se sont imposés comme des outils prometteurs pour la classification automatique des images médicales. Ces modèles, entraînés sur des ensembles de données diversifiés, ont démontré leur capacité à détecter avec précision des

anomalies complexes dans les images médicales. L'application de ces technologies aux images IRM ouvre la voie à des systèmes d'aide au diagnostic plus rapides, plus précis et plus accessibles, améliorant ainsi la prise en charge des patients.

Ce projet s'inscrit dans ce contexte en explorant l'utilisation des CNN pour classer automatiquement les tumeurs cérébrales à partir d'images IRM. En outre, il propose une comparaison rigoureuse entre un modèle personnalisé et des modèles pré-entraînés tels qu'ImageNet et InceptionV3 afin d'évaluer leurs performances respectives et leurs limites.

Problématique

Le diagnostic précis et rapide des tumeurs cérébrales est crucial pour garantir un traitement efficace et améliorer les perspectives de survie des patients. Cependant, les méthodes actuelles de diagnostic reposent sur des analyses manuelles des images IRM par des radiologues, ce qui est non seulement long mais également sujet à des variations inter- et intra-observateurs. Par ailleurs, l'augmentation exponentielle du volume de données médicales rend difficile une analyse rapide et exhaustive.

L'enjeu de ce projet est donc double :

1. Comment développer un modèle de classification automatisée capable d'identifier les tumeurs cérébrales à partir d'images IRM avec un haut degré de précision ?
2. Quels sont les avantages et limites des modèles pré-entraînés par rapport à un modèle conçu sur mesure pour cette tâche spécifique ?

Objectif

L'objectif principal est de pallier les limitations des approches traditionnelles et de fournir une solution basée sur l'intelligence artificielle qui, à notre humble niveau, puisse égaliser voire surpasser les performances de modèles préexistants.

Proposition d'approche

Pour répondre à l'objectif défini, nous proposons de réaliser un **benchmark des performances** en comparant un modèle préexistant et les modèles développés spécifiquement pour ce projet en utilisant TensorFlow. Cette approche nous permettra d'évaluer l'efficacité relative des solutions existantes et de nos propres modèles, tout en identifiant leurs forces et leurs limites respectives.

Dans ce cadre, nous avons choisi de fixer comme "baseline" un modèle de réseau neuronal développé par Google et bien connu pour l'analyse d'images médicales : **InceptionV3**. Bien qu'il ne fasse pas partie du benchmark, il servira de référence en terme de performances et d'axe d'amélioration.

Tous les modèles seront évalués à l'aide de métriques standard, telles que la précision, le rappel, le F1-score, l'AUC-ROC, ainsi que l'analyse de leurs matrices de confusion pour garantir une comparaison objective.

Cette démarche permettra non seulement de déterminer quel modèle offre les meilleures performances pour la classification des tumeurs cérébrales, mais également de mieux comprendre les compromis entre complexité des modèles, précision et généralisation. En outre, cette comparaison apportera des insights précieux sur les limites des solutions actuelles et guidera les futures améliorations, qu'on pourra évoquer dans le chapitre [Discussion](#).

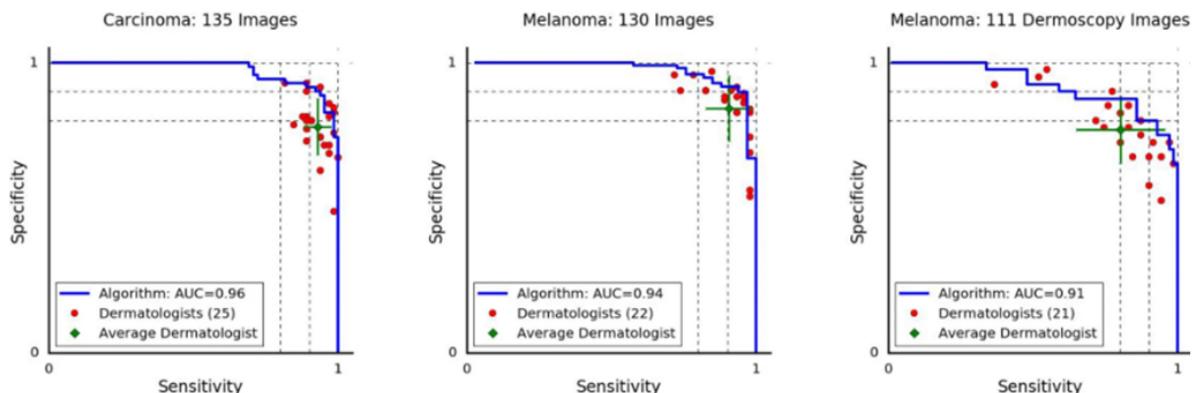
Chapitre 2

Revue de l'État de l'Art

Le domaine de l'imagerie médicale a connu de véritables avancées à l'aide du déploiement de l'intelligence artificielle, en particulier en apprentissage profond. Depuis l'introduction des réseaux neuronaux convolutionnels (CNNs) par [LeCun et al. \(1998\)](#), ces modèles sont devenus un pilier pour la reconnaissance d'images en raison de leur capacité à extraire des caractéristiques complexes à partir de données brutes. Les CNN ont été rapidement adoptés pour des applications médicales, notamment pour l'analyse des images IRM, des tomodensitogrammes (CT) et des mammographies.

Des modèles pré-entraînés, tels que ResNet [He et al. \(2016\)](#), et InceptionV3 [Szegedy et al. \(2016\)](#), ont été adaptés pour des tâches spécifiques en imagerie médicale, grâce à la technique du transfert d'apprentissage. Cette méthode permet de tirer parti de modèles initialement entraînés sur des bases de données larges comme ImageNet [Deng et al. \(2009\)](#) pour des ensembles de données médicaux souvent limités.

Les approches basées sur les CNN ont permis des avancées significatives dans la détection et la classification des maladies. On peut citer l'exemple de [Bossé et al. \(2017\)](#), qui ont démontré l'efficacité des réseaux profonds pour identifier des lésions dermatologiques cancéreuses dans des images de lésions dermatologiques, atteignant des niveaux de précision comparables à ceux des experts humains :



Deep Learning outperforms the average dermatologist at skin cancer classification using photographic and dermoscopic images, [Nature 542, 115–118] [Bossé et al. \(2017\)](#)

Les aires sous les courbes (AUC) pour les 3 classes de lésions cancéreuses sont en moyenne au dessus du diagnostic d'un expert clinicien en dermatologie, représentés en rouge.

Contexte spécifique à la classification des tumeurs cérébrales

La classification des tumeurs cérébrales est un cas d'application spécifique de l'analyse d'images IRM. Les tumeurs cérébrales sont souvent classées en catégories telles que bénignes ou malignes, ou selon des types spécifiques (par exemple, glioblastome, méningiome). Les bases de données comme le BraTS Challenge [Menze et al. \(2015\)](#) ont fourni des ensembles de données annotées permettant de développer et tester des modèles d'apprentissage automatique.

Les travaux récents se concentrent sur l'amélioration de la précision et de la généralisation des modèles. Par exemple, [Mohsen et al. \(2018\)](#) ont utilisé des CNN pour classifier des tumeurs cérébrales avec des performances élevées sur des données IRM. Cependant, les défis incluent la faible disponibilité des données dû à la confidentialité de ces données, la variabilité des images, les différentes coupes d'IRM, ainsi que le déséquilibre des classes dans les bases de données.

Analyse critique des travaux existants

Bien que les approches actuelles aient montré des résultats prometteurs, plusieurs limitations subsistent :

1. Les modèles pré-entraînés comme InceptionV3 ou ResNet ne sont pas spécifiquement optimisés pour des caractéristiques spécifiques aux images médicales, et ils nécessiteraient une personnalisation pour des performances optimales.
2. La plupart des études utilisent des ensembles de données relativement petits, limitant la robustesse et la généralisabilité des modèles.
3. Peu de travaux intègrent des mécanismes d'explicabilité pour aider les cliniciens à interpréter les résultats des modèles.

Ces limitations soulignent l'importance de développer des modèles personnalisés, adaptés aux spécificités des images IRM et intégrant des méthodes pour interpréter les décisions des modèles.

Résumé

Dans ce chapitre, nous avons exploré l'état de l'art sur la classification des images médicales, en explorant la littérature spécifique aux tumeurs cérébrales. Les CNN et les modèles pré-entraînés ont révolutionné ce domaine, mais des défis subsistent en termes de généralisation, d'interprétabilité et d'explicabilité. Ces constats justifient la pertinence de ce projet, qui vise à proposer un modèle de classification adapté pour la classification des tumeurs cérébrales.

Chapitre 3

Présentation du jeu de données

Le [Brain Tumor Classification \(MRI\)](#) Dataset par Bhuvaji et al. (2020), disponible sur Kaggle, est une ressource précieuse pour la classification des tumeurs cérébrales à partir d'images IRM. Ce jeu de données est une compilation de trois sources distinctes : [Figshare](#), le dataset [SARTAJ](#), et [Br35H](#). Il a été conçu pour faciliter la recherche et le développement de modèles de classification des tumeurs cérébrales en fournissant une base diversifiée d'images IRM étiquetées.

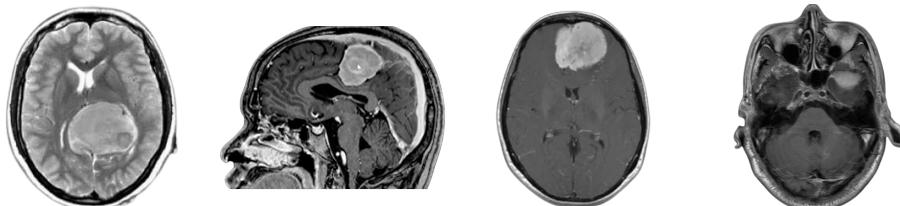
Le jeu de données comprend un total de 3 264 images IRM humaines, réparties en quatre catégories distinctes :

- **Gliome** : (926 images) Ressemble à une tâche floue dans le cerveau, car elles se mélangent souvent aux tissus environnants. Il est difficile de tracer des contours nets autour de cette tumeur. Elle peut avoir un centre sombre (zone morte) entouré d'une bordure plus claire, dûe au produit de contraste utilisé.



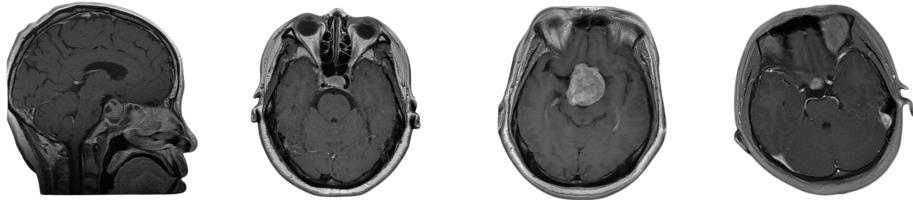
Exemples d'images IRM de gliomes

- **Méningiome** : (937 images) Ressemble à une boule bien définie attachée à la paroi externe du cerveau, comme si elle était "collée" à la surface. Elle est généralement bien délimitée et facile à repérer. Elle est homogène et brille uniformément après l'ajout d'un produit de contraste, ce qui la rend visuellement distincte.



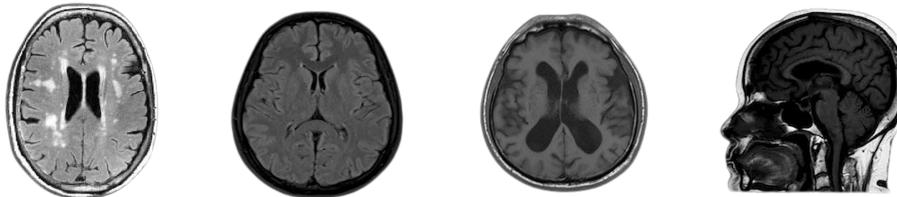
Exemples d'images IRM de méningiomes

- **Tumeur pituitaire** : (901 images) Ce type de tumeur se trouve au centre de la tête, dans une petite cavité sous le cerveau. Elle ressemble à une masse arrondie ou ovale. Elle peut pousser sur les tissus voisins, comme les nerfs liés à la vision, ce qui peut causer des symptômes spécifiques.



Exemples d'images IRM de tumeurs pituitaires

- **Aucune tumeur** : (500 images) Ces images montrent un cerveau normal sans anomalies visibles. Les structures cérébrales sont régulières, bien organisées et uniformes.



Exemples d'images IRM en l'absence de tumeur

Une difficulté majeure dans ce jeu de données réside dans la présence de différentes coupes en imagerie IRM. En effet, elles sont acquises selon plusieurs plans ou orientations, incluant :

- **Coupe axiale** : Vue de haut en bas (comme si l'on regardait la tête depuis le dessus)
- **Coupe sagittale** : Vue de profil (de gauche à droite ou inversement)
- **Coupe coronale** : Vue de face (comme si l'on regardait la tête de face)

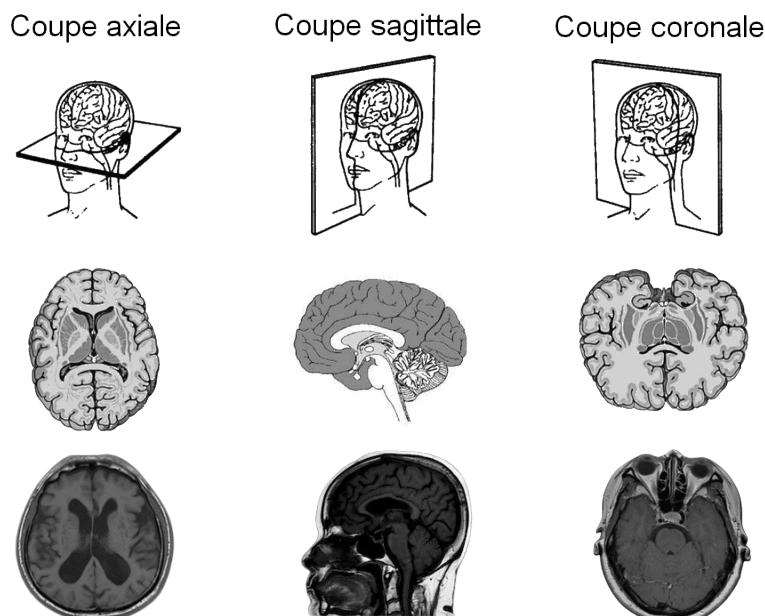


Illustration des différentes coupes en imagerie cérébrale

Les images sont déjà divisées en ensembles d'entraînement et de test, avec environ 12 % des images réservées pour le test et le reste pour l'entraînement. Chaque classe est représentée dans ces deux ensembles, permettant une évaluation rigoureuse des modèles de classification.

Il est également important de noter que les dimensions des images varient au sein du jeu de données. Par conséquent, une étape de prétraitement est nécessaire pour :

- Redimensionner les images à une taille uniforme.
- Supprimer les marges superflues autour des zones cérébrales pertinentes.

Chapitre 4

Data Visualisation

4.1 Preprocessing

Sur notre source Kaggle, les images sont déjà divisées en ensembles d'entraînement et de test, avec environ 88 % des images réservées pour l'entraînement et les 12% restants pour le test.

Cependant, comme nous voulons tester des modèles de réseaux convolutionnels (CNN), il est nécessaire de créer un ensemble de validation. En effet, le jeu de validation permettra d'estimer la capacité du modèle à généraliser au-delà des données d'entraînement. Sans cet ensemble, on pourrait risquer de confondre overfitting et réelle amélioration.

De plus, un prétraitement est nécessaire. Nous avons exploré les étapes de preprocessing effectuées implémentés dans le cadre du Brain Tumor Segmentation Challenge (BRaTS), un événement de référence dans la communauté IA, bénéficiant de douze ans d'expérience réussie dans la création de ressources pour la segmentation des tumeurs cérébrales. Parmi les approches proposées, la majorité sont assez complexes, enchaînant segmentation anatomique, corrections géométriques, normalisations avancées, et parfois même extraction de caractéristiques hand-crafted avant l'entraînement d'un modèle.

Cependant, des approches plus simples ont su montrer presque autant d'efficacité, comme le N4ITK de [Tustison et al. \(2010\)](#), correspondant à un prétraitement simpliste, axé sur la correction des inhomogénéités d'intensité et la normalisation, mais qui a largement prouvé son efficacité. Elle a démontré que le prétraitement minimal suffisait.

Nous avons donc choisi de pré-traiter notre jeu de données de la façon suivante :

- **Normalisation des images** : cette étape consiste à redimensionner les valeurs des pixels - dans notre cas en niveau de gris, comprises dans l'intervalle [0, 255] - afin que toutes les valeurs de pixels se situent dans l'intervalle [0, 1], selon la formule pixel normalisé = $\frac{\text{pixel} - \min(\text{pixel})}{\max(\text{pixel}) - \min(\text{pixel})}$
- **Standardisation des images** : cette étape consiste à ajuster les pixels pour qu'ils aient une moyenne de 0 et un écart-type de 1, selon la formule pixel standardisé = $\frac{\text{pixel} - \mu}{\sigma}$, où μ est la moyenne des pixels de l'image et σ leur écart-type. Ceci aide à accélérer la convergence lors de l'entraînement des modèles
- **Redimensionnement des images → 256×256 pixels** : cette étape permet d'uniformiser la taille des images pour qu'elles soient traitées efficacement par les modèles, en réduisant la complexité computationnelle, la vitesse d'entraînement. Elle assure également la compatibilité avec les différentes architectures de réseaux de neurones.

Une fois prétraitées, on crée l'ensemble de validation en isolant 20% des images de chaque classe de l'ensemble de test.

4.2 Équilibre des classes

Le dataset comprends 3 264 images IRM. Les images sont déjà divisées en ensemble d'entraînement (2 870) et de test (394). Nous allons extraire 20% des images de chaque classe de l'ensemble de test. Voici la répartition finale de chaque ensemble par classe :

Dans l'ensemble d'entraînement (2 297 images, correspondant à $\approx 71\%$ du dataset) :

- **Gliome** : 661 -> (vs 826 avant d'en extraire 20% pour la validation)
- **Méningiome** : 658 (vs 822)
- **Tumeur pituitaire** : 662 (vs 827)
- **Aucune tumeur** : 316 (vs 395)

Dans l'ensemble de validation (573 images, correspondant à $\approx 17\%$ du dataset) :

- **Gliome** : 165
- **Méningiome** : 164
- **Tumeur pituitaire** : 165
- **Aucune tumeur** : 79

L'ensemble de test, lui, demeure inchangé (394 images, correspondant à $\approx 12\%$ du dataset) :

- **Gliome** : 100
- **Méningiome** : 115
- **Tumeur pituitaire** : 74
- **Aucune tumeur** : 105

4.3 Représentation graphique des données

Voici une représentation graphique de la répartition des images par classe :

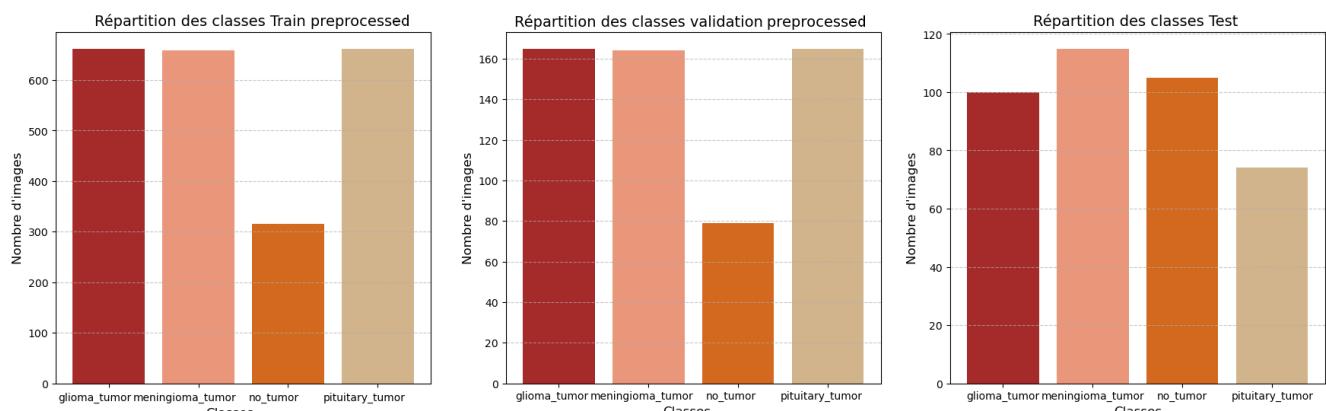


Diagramme à bâtons de la répartition des images par classe

On observe un équilibre plutôt bon entre les classes dans l'ensemble de test, en remarquant toutefois une légère sous-représentation de la classe "Pas de tumeur" dans l'ensemble d'entraînement.

4.4 Analyse de la variance

Pour évaluer la dispersion et la séparabilité des données, nous avons effectué une analyse des variances. Ce choix s'explique par la nature particulière des IRM, où des coupes différentes peuvent présenter des similarités visuelles importantes, même entre des classes distinctes. En calculant les variance intra-classes et inter-classes, nous voulons vérifier si les données sont suffisamment bien séparées afin d'estimer la complexité de la tâche de classification. En effet une séparabilité élevée garantit que le modèle aura moins de difficulté à distinguer chaque classe.

Pour ce faire, les images sont aplatis et converties en vecteurs unidimensionnels (1D) via la fonction `np.array(img).flatten`, facilitant ainsi les calculs statistiques nécessaires.

4.4.1 Variance intra-classes

La variance intra-classes mesure la dispersion des échantillons au sein d'une classe k par rapport à sa moyenne μ_k . La variance intra-classes σ_{intra}^2 est calculée comme suit :

$$\sigma_{\text{intra}}^2 = \frac{1}{n_k} \sum_i^{n_k} \|x_i - \mu_k\|^2$$

où :

- x_i est le vecteur représentant une image de la classe k
- n_k est le nombre d'images dans la classe

En voici les résultats :

- **Gliome** : 178 346 470.0735555
- **Méningiome** : 330 496 060.5393776
- **Tumeur pituitaire** : 231 571 516.7505451
- **Aucune tumeur** : 484 557 952.5108772

Bien que ces chiffres semblent élevés (ordre de 10^8), pour des jeux de données d'images comportant plusieurs centaines ou milliers de pixels, des variances, cet ordre de grandeur est typiques. De plus cela s'explique aussi par la diversité des coupes présentes dans une même classe ou l'intensité des IRM qui peut être différente d'une image à l'autre. L'orientation de la boîte crânienne et sa dimension influent également.

4.4.2 Variance inter-classes

La variance inter-classes mesure la dispersion mesurée la dispersion des moyennes des classes μ_k par rapport à la moyenne globale μ . La variance inter-classes σ_{inter}^2 est calculée comme suit :

$$\sigma_{\text{inter}}^2 = \sum_k n_k \|\mu_k - \mu\|^2 \quad \text{où } n_k \text{ est la dimension de la classe}$$

La variance inter-classe de notre jeu de donnée est : 50 487 799 049.60.

4.4.3 Ratio inter/intra-classes

Pour évaluer globalement la qualité des données, nous utilisons le ratio inter/intra-classes défini comme suit :

$$R = \frac{\sigma_{\text{inter}}^2}{\sigma_{\text{intra-global}}^2}$$

où :

- $\sigma_{\text{intra-global}}^2 = \sum_k^K \sigma_{\text{intra}_k}^2$
- K le nombre de classes

Avec nos données, cela donne :

$$R = \frac{50487799049.60}{178346470.0735555 + 330496060.5393776 + 231571516.7505451 + 484557952.5108772} \approx 41.22$$

Interprétation : Un ratio supérieur à 1 indique une bonne séparabilité des classes. Ici, un ratio de 41.22 démontre une excellente séparabilité, ce qui suggère que la tâche de classification sera relativement moins complexe, car les classes sont bien distinctes.

4.5 Architecture et Arborescence du projet

Dans un soucis de rendre nos modèles les plus reproductibles possibles, nous avons tenu à suivre quelques bonnes pratiques en informatique, lorsqu'il s'agit de développements de modèles de Deep Learning. Cela facilitera l'onboarding de nouveaux contributeurs et la maintenance du projet sur le long terme. Les pratiques suivies et l'arborescence sont consultables en [Annexe A : Bonnes pratiques et Arborescence du projet](#).

Chapitre 5

Métriques utilisées pour le benchmarking

Avant d'introduire et présenter nos modèles, leur fonctionnement et leurs résultats, il est important d'identifier les métriques que nous utiliseront pour évaluer et comparer nos modèles. Commençons par les définir :

5.1 Définition des Métriques

Précision La précision mesure la proportion des prédictions positives correctes pour une classe donnée par rapport à toutes les prédictions positives faites pour cette classe :

$$\text{Précision}_i = \frac{VP_i}{VP_i + FP_i},$$

où :

- VP_i (Vrais Positifs) : Nombre d'échantillons de la classe i correctement prédits.
- FP_i (Faux Positifs) : Nombre d'échantillons appartenant à d'autres classes mais incorrectement prédits comme appartenant à i .

Rappel Le rappel mesure la proportion des échantillons réellement appartenant à une classe qui ont été correctement identifiés :

$$\text{Rappel}_i = \frac{VP_i}{VP_i + FN_i},$$

où :

- FN_i (Faux Négatifs) : Nombre d'échantillons de la classe i mal prédits comme appartenant à d'autres classes.
- VP_i (Vrais Positifs) : Nombre d'échantillons de la classe i correctement prédits.
- FP_i (Faux Positifs) : Nombre d'échantillons appartenant à d'autres classes mais incorrectement prédits comme appartenant à i .

F_1 -Score Le F_1 -score combine la précision et le rappel dans une moyenne harmonique, particulièrement utile en cas de données déséquilibrées :

$$F_{1,i} = 2 \cdot \frac{\text{Précision}_i \cdot \text{Rappel}_i}{\text{Précision}_i + \text{Rappel}_i}.$$

AUC (Area Under the Curve) L'AUC mesure l'aire sous la courbe ROC (Receiver Operating Characteristic), qui trace le taux de vrais positifs (TPR_i) en fonction du taux de faux positifs (FPR_i) à différents seuils de décision :

$$AUC_i = \int_0^1 TPR_i(FPR_i) d(FPR_i),$$

où :

- $TPR_i = \frac{VP_i}{VP_i + FN_i}$: Taux de vrais positifs pour la classe i .
- $FPR_i = \frac{FP_i}{FP_i + TN_i}$: Taux de faux positifs pour la classe i .
- TN_i (Vrais Négatifs) : Nombre d'échantillons correctement identifiés comme n'appartenant pas à la classe i .

5.2 Métriques Globales

Pour comparer les modèles de manière globale, nous allons définir des métriques agrégées basées sur une moyenne pondérée pour toutes les classes. C'est les métriques que nous utiliserons pour dresser le tableau final de notre benchmark.

Précision Globale (Moyenne Pondérée)

$$\text{Précision}_{\text{Globale}} = \frac{\sum_{i=1}^4 n_i \cdot \text{Précision}_i}{\sum_{i=1}^4 n_i}$$

où :

- n_i : Nombre d'échantillons dans la classe i
- Précision_i : Précision pour la classe i

Rappel Global (Moyenne Pondérée)

$$\text{Rappel}_{\text{Global}} = \frac{\sum_{i=1}^4 n_i \cdot \text{Rappel}_i}{\sum_{i=1}^4 n_i}$$

F_1 -Score Global (Basé sur les Totaux Agrégés)

$$F_{1,\text{Global}} = 2 \cdot \frac{\text{Précision}_{\text{Globale}} \cdot \text{Rappel}_{\text{Global}}}{\text{Précision}_{\text{Globale}} + \text{Rappel}_{\text{Global}}}$$

AUC Global (Moyenne Pondérée)

$$\text{AUC}_{\text{Global}} = \frac{\sum_{i=1}^4 n_i \cdot \text{AUC}_i}{\sum_{i=1}^4 n_i}$$

où n_i est le nombre d'échantillons dans la classe i et AUC_i est l'AUC pour la classe i

Ces métriques agrégées permettent de comparer les modèles en tenant compte des performances pour toutes les classes.

Chapitre 6

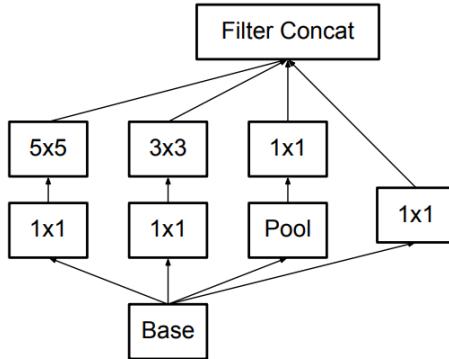
Modèles

6.1 Baseline model : Inception V3

InceptionV3 est une architecture de réseau de neurones convolutifs (CNN) développée par Google, introduite dans le papier "Rethinking the Inception Architecture for Computer Vision" de He et al. (2016). C'est une amélioration des anciennes versions de Inception, en terme de performance et d'efficacité computationnelle. Celui-ci a l'avantage non négligeable d'être pré-entraîné sur ImageNet, contenant plus d'un million d'images et 1 000 classes, fournissant ainsi un ensemble de poids initiaux qui ont déjà appris une grande variété de patrons visuels.

Ses modules permettent de capturer à la fois des caractéristiques générales et locales dans une image. Cela est possible en partie grâce à la combinaison de plusieurs convolutions de différentes tailles de noyaux dans une même couche :

- Convolution 1x1 qui permet de réduire le nombre de canaux et d'extraire des caractéristiques fines
- Convolutions 3x3 et 5x5 qui permet de capturer des motifs locaux plus généraux et plus complexes
- Pooling qui permet de réduire la résolution et extraire des informations invariantes aux translations



Original Inception introduit par Szegedy et al. (2014)

Ses améliorations

InceptionV3 présente plusieurs améliorations par rapport à ses versions antérieures :

- Les convolutions de grande taille (par exemple, 5x5) sont remplacées par une séquence de convolutions plus petites (par exemple, deux convolutions 3x3). Cela réduit le coût computationnel tout en préservant la capacité à capturer des motifs complexes.
- L'intégration dans plusieurs couches de la batch normalisation pour accélérer la convergence et stabiliser l'apprentissage.
- Des prédicteurs auxiliaires sont utilisés à des étapes intermédiaires pour améliorer la rétro propagation des gradients (observable plutôt vers la fin de l'entraînement) et réduire le risque de sur-apprentissage.

Les résultats de Szegedy et al. (2014) sont comparés avec les meilleures performances publiées d'inférence en ensemble sur le benchmark de classification ILSVRC 2012 dans le tableau suivant :

Network	Models Evaluated	Crops Evaluated	Top-1 Error	Top-5 Error
VGGNet [18]	2	-	23.7%	6.8%
GoogLeNet [20]	7	144	-	6.67%
PReLU [6]	-	-	-	4.94%
BN-Inception [7]	6	144	20.1%	4.9%
Inception-v3	4	144	17.2%	3.58%*

Résultats d'évaluation en ensemble de He et al. (2016) sur le benchmark ILSVRC 2012

InceptionV3 affiche des performances avec un taux d'erreur Top-1 de 17.2% et Top-5 de 3.58%, surpassant VGGNet (23.7% et 6.8%) et GoogLeNet (6.67% en Top-5). Sa faible erreur Top-5 démontre sa capacité à identifier avec précision les classes dans un ensemble de données large et diversifié comme ImageNet. Cela en fait un modèle de référence pour la classification d'images.

Structure globale

L'architecture InceptionV3 comprend environ 24 millions de paramètres répartis en plusieurs blocs. Elle peut être divisée en trois grandes parties :

- Extraction des caractéristiques de bas niveau : Les premières couches extraient des caractéristiques simples à partir des pixels bruts.
- Extraction des caractéristiques complexes : Les modules Inception permettent d'extraire des motifs de plus en plus complexes et invariants, grâce à des convolutions à différentes échelles.
- Classification finale : Une couche GlobalAveragePooling suit les blocs convolutifs, compressant l'information en un vecteur de caractéristiques. Ce vecteur est ensuite transmis à des couches denses pour effectuer la classification finale.

Implémentation du modèle

Paramètres utilisés :

- `weights = 'imagenet'` on utilise les poids pré-appris sur ImageNet
- `include_top = False` : on exclut les couches finales de classification du modèle pré-entraîné. Le modèle ne produit plus de prédictions sur les classes d'ImageNet, mais seulement des cartes de caractéristiques extraites par les couches convolutives précédentes.

Couches ajoutées à la base Inception :

- **GlobalAveragePooling2D** : Comprime les caractéristiques spatiales extraites par InceptionV3 en un vecteur de caractéristiques globales pour chaque image.
- **Dense (256 unités)** : Une couche entièrement connectée avec une activation ReLU, introduisant de la non-linéarité pour apprendre des relations complexes entre les caractéristiques.
- **Batch Normalization** : Normalise les activations pour accélérer l'apprentissage et réduire les risques de divergence.
- **Dropout (0.5)** : Désactive aléatoirement 50% des neurones lors de chaque itération pour limiter le sur-apprentissage.
- **Dense (128 unités)** : Une seconde couche dense avec la même configuration que la précédente, permettant d'affiner les représentations apprises.
- **Dense (4 unités)** : Une couche finale avec activation softmax, produisant des probabilités pour les 4 classes de tumeurs.

Callbacks pour l'entraînement :

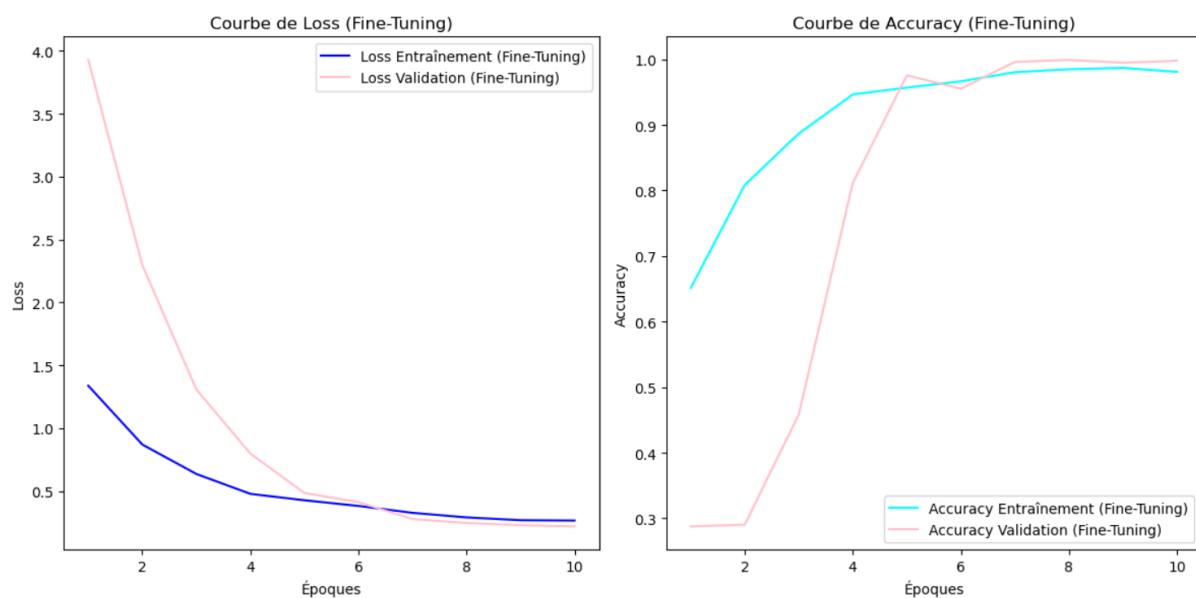
- **ReduceLROnPlateau** : Réduit le taux d'apprentissage lorsque la perte de validation ne diminue plus, pour permettre au modèle de faire de plus petits ajustements et continuer à s'améliorer.
- **EarlyStopping** : Arrête l'entraînement si la perte de validation ne s'améliore pas pendant 5 époques consécutives, permettant d'éviter un surapprentissage inutile.

Fine-tuning

Après l'entraînement initial des couches personnalisées, certaines couches de la base InceptionV3 sont dégelées pour ajuster leurs poids au dataset spécifique. Le fine-tuning est effectué avec :

- Déblocage partiel des couches : Les 200 premières couches d'InceptionV3 restent gelées pour conserver les caractéristiques générales, tandis que les couches plus profondes s'ajustent aux données spécifiques.
- Learning rate réduit (10^{-4}) : Pour éviter de perturber excessivement les poids pré-appris

Courbes Loss et Accuracy



Évolution des courbes Loss et Accuracy du modèle sur les ensembles d'entraînement et de validation pour le modèle InceptionV3

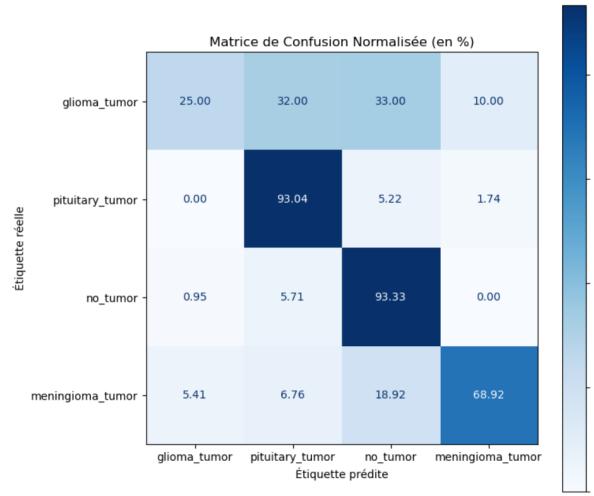
Interprétation des courbes

La perte d'entraînement décroît de manière régulière, passant d'environ 4.0 à moins de 0.5, ce qui reflète un ajustement efficace des paramètres du modèle aux données. La perte de validation suit une tendance similaire, convergeant vers une valeur proche de 0.5, ce qui indique une bonne généralisation sur l'ensemble de validation.

L'exactitude d'entraînement atteint 98%, ce qui montre que le modèle est capable de s'adapter parfaitement aux données d'entraînement. L'exactitude de validation augmente rapidement, dépassant 90% après quelques époques et se stabilisant par la suite autour des 99% ce qui est un indicateur de bonnes performances générales sur des données inconnues.

Ces résultats indiquent que le modèle a réussi à tirer profit des caractéristiques pré-apprises de InceptionV3 tout en s'adaptant efficacement à la tâche spécifique.

Matrice de confusion



Matrice de confusion du modèle InceptionV3

Interprétation

La matrice de confusion montre la répartition des prédictions du modèle par rapport aux véritables étiquettes des classes. Voici quelques observations notables :

- Classe **Gliome** : Le modèle montre une performance relativement faible sur cette classe, avec une exactitude de 25%. Cela pourrait être dû à des similitudes visuelles entre cette classe et les autres.
- Classe **Pituitaire** : Les performances sont bonne, avec une précision de 93%. Cela suggère que les caractéristiques de cette classe sont distinctives et bien captées par le modèle.
- Classe **Pas de tumeur** : Une précision de 93% est également observée, indiquant que le modèle distingue bien les cas sans tumeur des autres classes.
- Classe **Méningiome** : Une précision de 68% est obtenue, avec un confusion notable avec la classe "Pas de tumeur".

Voici un tableau qui dresse les performances de InceptionV3 :

Modèle	Activation	Nb couches	Pooling	Data Aug	Nb d'epochs	Performances globales						
						Acc	Loss	Prec	Recall	F1	FP	FN
InceptionV3	ReLU	InceptionV3 + 3 couches	Global Average	Non	15	0.71	1.41	0.74	0.71	0.68	123	123
Performances sur la Classe Gliome							0.91	0.5	0.65	7	75	
Performances sur la Classe Méningiome							0.77	0.50	0.61	11	36	
Performances sur la Classe Pas de tumeur							0.50	0.89	0.64	59	7	
Performances sur la Classe Pituitaire							0.50	0.90	0.64	46	5	

Table 6.1 : Résumé des performances du modèle InceptionV3

Résumé du modèle InceptionV3

Les bonnes performances globales, avec une exactitude de validation supérieure à 90%, montrent que le fine-tuning a permis au modèle d'apprendre des caractéristiques adaptées aux données. Les confusions observées dans la matrice de confusion (notamment entre Gliome et les autres classes) peuvent être dues à :

- Une similitude visuelle entre certaines classes.
- Un déséquilibre dans les données d'entraînement.

De plus, au vu du contexte clinique et notre problématique qui est de réduire au mieux le nombre de faux négatifs, ce modèle reste à améliorer pour obtenir des résultats satisfaisants dans ce contexte.

6.2 Modèle ReLU

Définition de la fonction d'activation :

$$f(x) = \max(0, x) = \begin{cases} x & \text{si } x > 0, \\ 0 & \text{si } x \leq 0. \end{cases}$$

Les avantages et inconvénients de ReLU (d'après [cette vidéo](#) d'E. Scornet) sont listés ci-dessous :

Avantages :

- Simplicité : Facile à calculer
- Évite les saturations : Contrairement à Sigmoid ou Tanh, ReLU ne sature pas pour les grandes valeurs positives, ce qui accélère la convergence
- Spécificité : Active uniquement une partie des neurones ($x > 0$), rendant le modèle plus efficace

Inconvénients : Problème de "neurones morts" : Les valeurs négatives entraînent une sortie constante de 0, ce qui peut désactiver certains neurones de manière permanente.

Fonction de perte : Sparse Categorical Crossentropy

La fonction de perte mesure l'écart entre les prédictions du modèle et les vraies étiquettes. Elle guide l'entraînement en ajustant les poids du modèle pour minimiser cet écart.

Formule de la Sparse Categorical Crossentropy :

$$-\frac{1}{N} \sum_{i=1}^N \log(p_{i,y_i}),$$

où :

- N est le nombre d'échantillon total
- y_i représente la classe correcte (indice) pour la $i^{\text{ème}}$ image
- p_{i,y_i} est la probabilité prédite pour la classe correcte y_i , sortie par le modèle

Particularité : Les étiquettes sont sparses c'est-à-dire que ce sont des entiers représentant les classes (0, 1, 2, ...), au lieu d'un vecteur one-hot encodé.

Justification de l'utilisation de cette fonction de perte :

- Classification multicatégorie : Idéal pour des problèmes où les classes sont exclusives (par ex., une image appartient à une seule catégorie)
- Évaluation probabiliste : Utilise la sortie de la couche softmax pour attribuer des probabilités à chaque classe

Architecture du modèle :

Entrée du modèle : Images $256 \times 256 \times 1$ (grayscale)

- 4 couches de convolution avec activation ReLU de noyaux 3x3
Nombre de filtres : 32, 64, 128, 128
- 2 couches de max pooling (2x2) pour la réduction dimensionnelle
- 1 couche fully connected avec 64 neurones avec activation ReLU
- 1 couche de Dropout (20% de neurones désactivés pour éviter le sur-apprentissage)

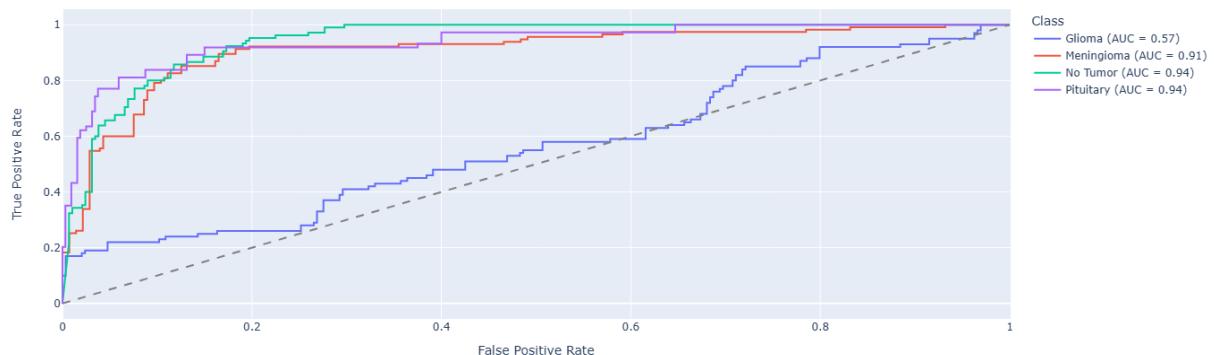
Sortie du modèle : 4 neurones avec activation softmax (classification multicatégorie)

Paramètres d'entraînement :

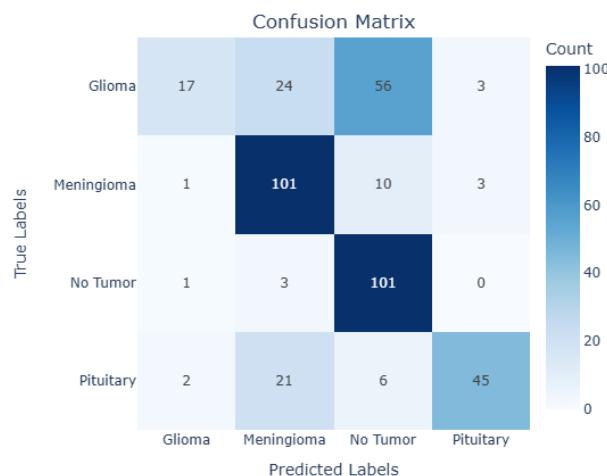
- **Optimiseur** : Adam (learning rate initial : 0.001)
- **Fonction de perte** : Sparse categorical crossentropy
- Metric : Accuracy
- Batch size : 32
- Validation split : 20% des données d'entraînement réservées à la validation
- Nombre d'epochs : 6

Particularité : Sans augmentation de données, les données d'entraînement sont donc utilisées telles quelles, sans modification ni équilibrage.

Courbe ROC



Matrice de confusion



Voici un tableau qui dresse les performances du modèle ReLU :

Modèle	Activation	Nb couches	Max pooling	Data Aug	Nb d'epochs	Performances globales							
						Acc	Loss	Prec	Recall	F1	FP	FN	AUC
ReLU	ReLU	4	2	Non	6	0.67	4.16	0.72	0.67	0.62	130	130	0.84
Performances sur la Classe Gliome							0.81	0.17	0.28	4	83	0.84	
Performances sur la Classe Méningiome							0.68	0.88	0.77	48	14	0.91	
Performances sur la Classe Pas de tumeur							0.58	0.96	0.73	72	4	0.94	
Performances sur la Classe Pituitaire							0.88	0.61	0.72	6	29	0.94	

Table 6.2 : Résumé des performances du modèle ReLU, sans Data Augmentation

6.3 Modèle Augmented ReLU

Architecture du modèle : Identique au modèle précédent ReLU.

Paramètres d'entraînement : Identiques à ceux du modèle précédent ReLU.

Particularité : On effectue une augmentation de données, que nous allons décrire ci-dessous.

- Équilibrage des classes : Ajout d'images synthétiques pour les classes sous-représentées.
- Transformations appliquées : Zoom, décalage et autres opérations pour enrichir les données et améliorer la robustesse du modèle.

Équilibre des classes : Data Augmentation avec ImageDataGenerator

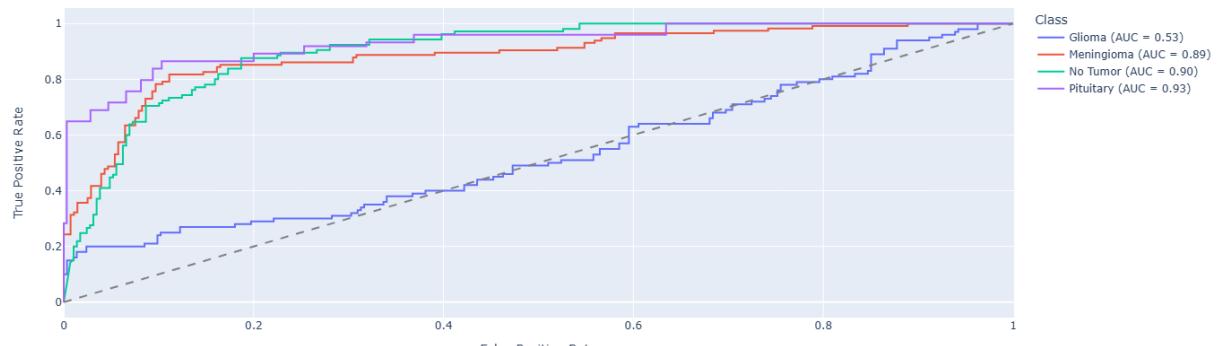
Qu'est-ce que ImageDataGenerator ? L'ImageDataGenerator est une classe de la bibliothèque Keras utilisée pour générer de nouvelles images à partir des données existantes. Elle applique des transformations comme des rotations, des zooms ou des décalages afin d'augmenter artificiellement la taille et la diversité du jeu de données.

Transformations appliquées Pour équilibrer les classes et améliorer la robustesse du modèle, nous avons utilisé les transformations suivantes :

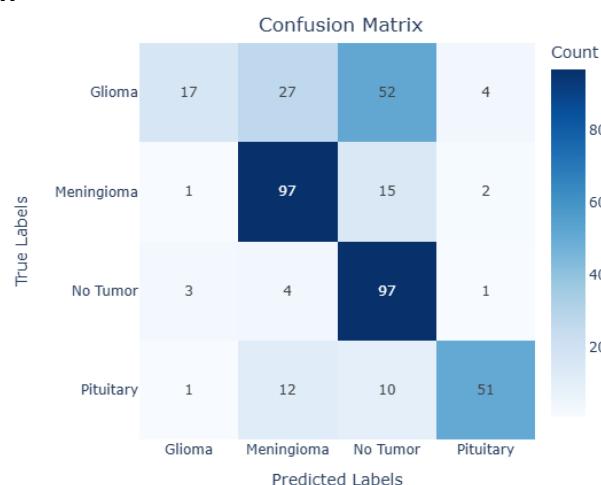
- **Rotations** : Pour varier les orientations des images
- **Zooms** : Pour simuler des variations de taille ou d'échelle
- **Décalages horizontaux et verticaux** : Pour introduire des variations dans les positions

Objectifs : Cette technique a pour objectif de générer plus d'exemples pour les classes sous-représentées et d'améliorer la robustesse du modèle afin de le rendre plus résilient aux variations des données du monde réel.

Courbe ROC



Matrice de confusion



Voici un tableau qui dresse les performances du modèle Augmented ReLU :

Modèle	Activation	Nb couches	Max pooling	Data Aug	Nb d'epochs	Performances globales							
						Acc	Loss	Prec	Recall	F1	FP	FN	AUC
ReLU	ReLU	4	2	Oui	6	0.66	4.79	0.62	0.66	0.62	132	132	0.81
Performances sur la Classe Gliome						0.77	0.17	0.28	5	83	0.53		
Performances sur la Classe Méningiome						0.69	0.84	0.76	43	18	0.89		
Performances sur la Classe Pas de tumeur						0.56	0.92	0.70	77	8	0.90		
Performances sur la Classe Pituitaire						0.88	0.69	0.77	7	23	0.93		

Table 6.3 : Résumé des performances du modèle ReLU, avec Data Augmentation

6.4 Modèle Sigmoid

6.4.1 Implémentation du modèle

Architecture du modèle :

Entrée du modèle : Images $256 \times 256 \times 1$ (grayscale)

- 4 couches de convolution avec activation Sigmoid de noyaux 3×3
Nombre de filtres : 32, 64, 128, 128
- 2 couches de max pooling (2×2) pour la réduction dimensionnelle
- 1 couche fully connected avec 64 neurones avec activation ReLU
- 1 couche de Dropout (20% de neurones désactivés pour éviter le sur-apprentissage)

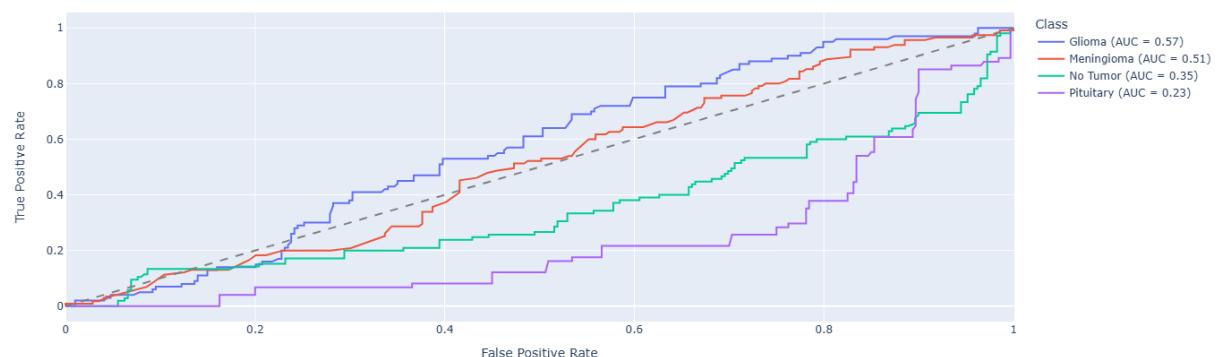
Sortie du modèle : 4 neurones avec activation softmax (classification multicatégorie)

Particularité : Utilisation de l'activation sigmoïd dans les couches de convolution, contrairement aux modèles précédents qui utilisent ReLU.

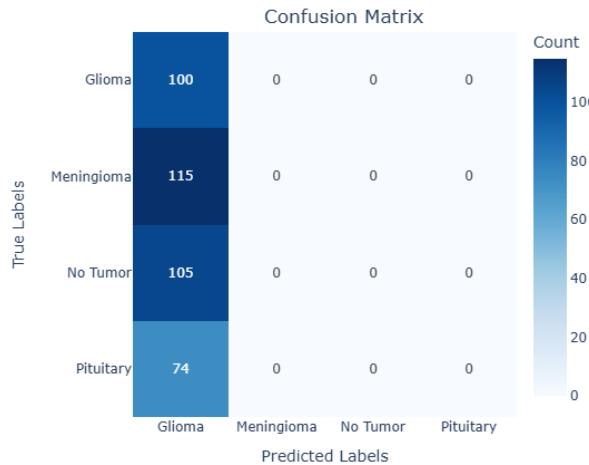
Paramètres d'entraînement :

- **Optimiseur** : Adam (learning rate initial : 0.001)
- **Fonction de perte** : Sparse categorical crossentropy
- Metric : Accuracy
- Batch size : 32
- Validation split : 20% des données d'entraînement réservées à la validation
- Nombre d'epochs : 6

Courbe ROC



Matrice de confusion



Voici un tableau qui dresse les performances du modèle Sigmoid :

Modèle	Activation	Nb couches	Max pooling	Data Aug	Nb d'epochs	Performances globales							
						Acc	Loss	Prec	Recall	F1	FP	FN	AUC
Sigmoid	Sigmoid	4	2	Non	6	0.25	1.43	0.06	0.25	0.10	294	294	0.42
Performances sur la Classe Gliome							0.25	1.00	0.40	5	83	0.57	
Performances sur la Classe Méningiome							0.00	0.00	0.00	0	115	0.51	
Performances sur la Classe Pas de tumeur							0.00	0.00	0.00	0	105	0.35	
Performances sur la Classe Pituitaire							0.00	0.00	0.00	0	74	0.23	

Table 6.4 : Résumé des performances du modèle Sigmoid

Interprétation des résultats

Les résultats obtenus sur la loss et l'accuracy montrent une stagnation des performances du modèle et une perte qui cesse de diminuer rapidement après les premières itérations. La matrice de confusion révèle une prédiction systématique de la même classe, indiquant une incapacité du modèle à différencier les classes. Ce comportement peut être attribué à l'utilisation de la fonction d'activation sigmoïde dans les couches convolutives, dont l'effet principal est de provoquer une saturation des gradients. La fonction sigmoïde, définie par :

$$\rho(x) = \frac{\exp(x)}{(1 + \exp(x))}$$

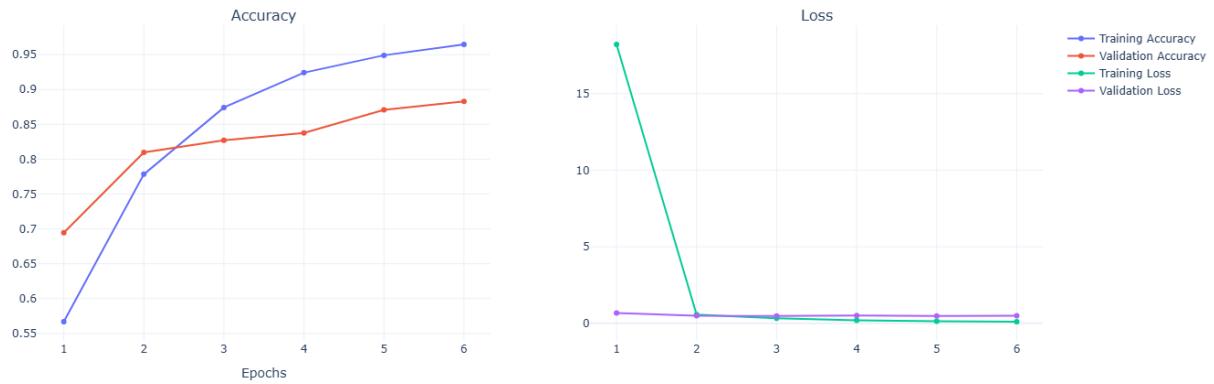
La fonction sigmoïde sature lorsque ses entrées sont très grandes ou très petites (loin de 0), rendant sa dérivée proche de zéro. Dans ces zones saturées, les gradients deviennent extrêmement faibles (**phénomène de vanishing gradients**), empêchant une mise à jour efficace des poids, notamment dans les couches profondes. Cette saturation peut être causée par plusieurs facteurs. Tout d'abord, une initialisation aléatoire des poids peut produire des activations très grandes ou très petites dès le début, poussant les sorties de la sigmoïde dans ses zones saturées.

De plus, l'accumulation des transformations non linéaires à travers les couches successives amplifie ce phénomène, rendant les activations encore plus éloignées de la plage efficace de la sigmoïde. Ces facteurs combinés bloquent la rétropropagation des gradients dans les couches profondes et empêchent le modèle d'apprendre efficacement. Cette limitation bloque l'apprentissage des caractéristiques discriminantes nécessaires à la classification et conduit le modèle à favoriser systématiquement la classe majoritaire, minimisant ainsi globalement la perte sans réellement apprendre.

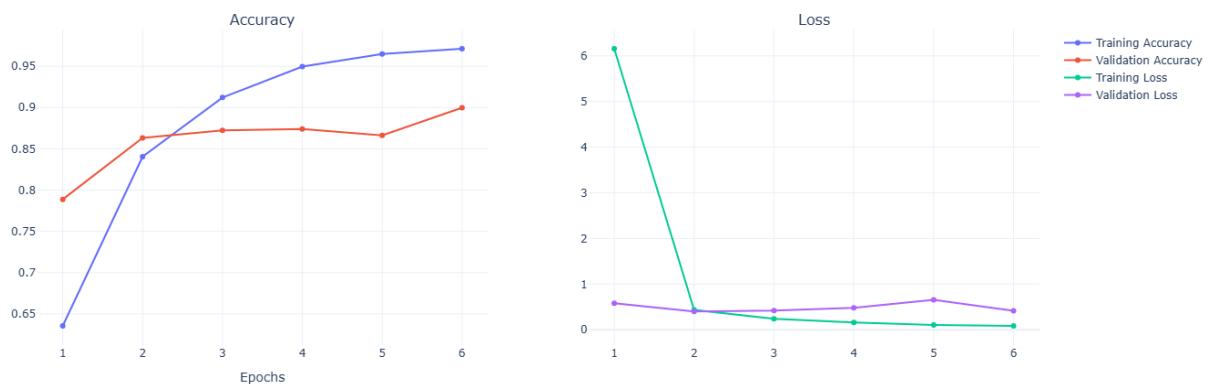
Chapitre 7

Resultats et Benchmark

7.1 ReLU vs Augmented ReLU



Courbes Accuracy et Loss du modèle ReLU sur les ensembles d'entraînement et de validation



Courbes Accuracy et Loss du modèle Augmented ReLU sur les ensembles d'entraînement et de validation

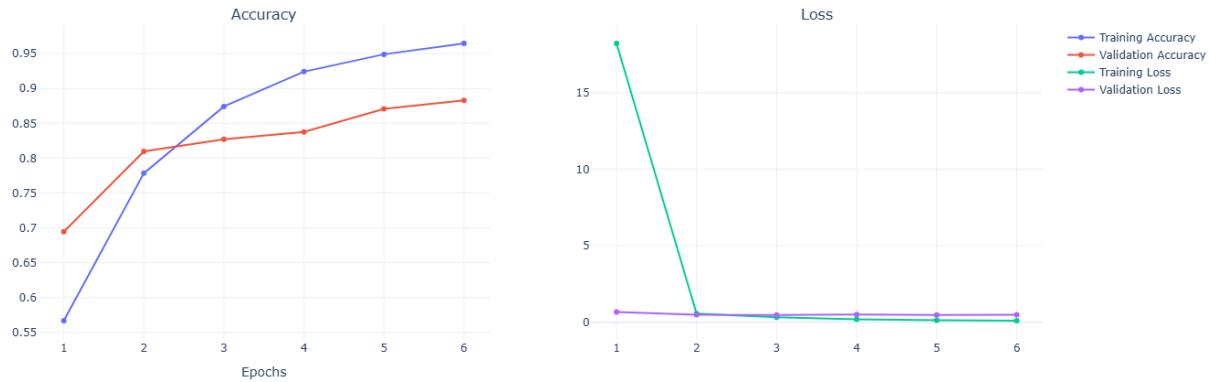
Les courbes de Loss et d'Accuracy semblent similaires pour les deux modèles étudiés. Afin d'approfondir notre comparaison, analysons leurs performances sur le jeu de test : On obtient 0.6701 pour ReLU vs 0.664 pour Augmented ReLU. Les tests d'accuracy sont quasi équivalents. Cela indique que l'équilibre des classes et l'augmentation des données **n'apportent pas d'amélioration significative**.

7.2 ReLU vs Sigmoid

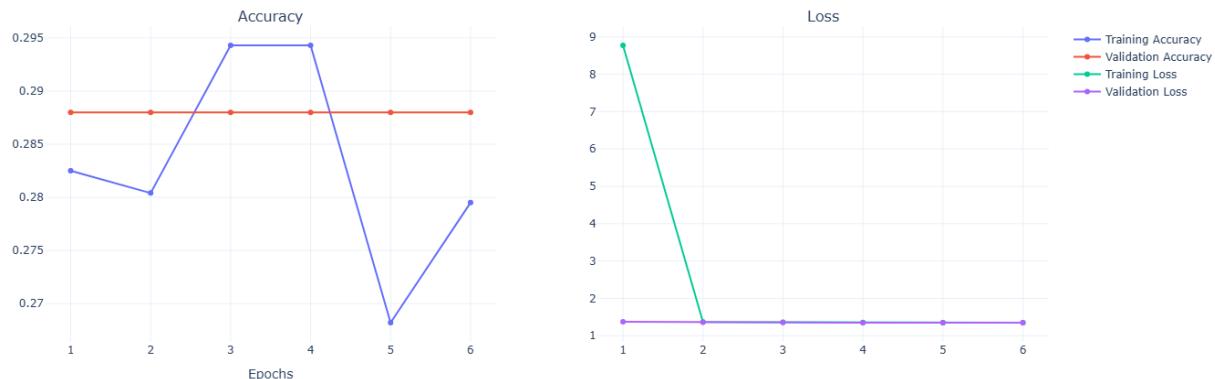
Comparaison des Fonctions d'activation

- **ReLU** : pour introduire de la non-linéarité et éviter les problèmes de gradient nul.
- **Sigmoid** : pour normaliser les sorties entre 0 et 1, souvent utilisée pour des tâches de classification binaire.

Affichons quelques courbes :



Courbes Accuracy et Loss du modèle ReLU sur les ensembles d'entraînement et de validation



Courbes Accuracy et Loss du modèle Sigmoid sur les ensembles d'entraînement et de validation

L'accuracy du modèle ReLU semble avoir une meilleure évolution au fil des epochs, et les courbes Loss ne permettent pas de les départager. Afin d'approfondir notre comparaison, analysons leurs performances sur le jeu de test : On obtient 0.6701 pour ReLU vs 0.2538 pour Sigmoid. Il n'y a plus de doute, **le modèle ReLU est bien meilleur que Sigmoid**.

Voici un barplot résumant le score par métrique et par modèle :

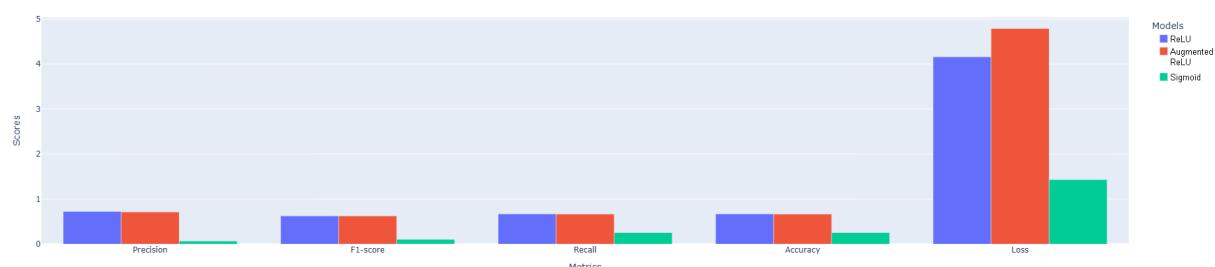


Diagramme en bâtons des métriques selon les différents modèles

7.3 Deux types de lecture

Lorsqu'on évalue les performances de modèles de classification dans un contexte médical, deux lectures distinctes peuvent être adoptées en fonction des priorités du projet. Nous allons définir les approches "Optimisation de la Performance Globale de Classification" et "Minimisation des Risques Cliniques".

7.3.1 Optimisation de la Performance Globale de Classification

Dans cette première lecture, l'objectif est de maximiser la performance globale du modèle, c'est-à-dire sa capacité à prédire correctement la classe de chaque échantillon, quelle que soit la classe. Les métriques telles que la précision, le rappel, le F1-score et l'accuracy globale sont utilisées pour évaluer le modèle. Cette méthode est particulièrement pertinente lorsque toutes les classes ont un poids équivalent en termes de conséquences, et que chaque erreur de classification a un impact similaire. Ce qui est questionable dans un contexte médical.

Cependant, en recherche ou dans des domaines où l'exactitude globale du modèle est évaluée, cette approche est souvent privilégiée. Par exemple, pour des applications dans lesquelles le modèle doit équilibrer les erreurs entre plusieurs catégories (comme dans la reconnaissance d'objets en vision par ordinateur ou les applications industrielles), l'optimisation globale offre une vue d'ensemble claire des performances du modèle.

Métriques utilisées : Cette approche repose sur des mesures standard telles que l'accuracy et le F1-score. Le F1-score, en particulier, offre un compromis équilibré entre précision et rappel, et est souvent utilisé lorsque les classes sont déséquilibrées.

Dans ce cadre, un faux positif et un faux négatif sont considérés comme ayant des conséquences similaires.

Voici la table récapitulative des performances pour nos 3 modèles, selon cette première lecture :

Modèle	Activation	Nb couches	Max pooling	Data Aug	Nb d'epochs	Performances globales							
						Acc	Loss	Prec	Recall	F1	FP	FN	AUC
ReLU	ReLU	4	2	Non	6	0.67	4.16	0.72	0.67	0.62	130	130	0.84
Augmented ReLU	ReLU	4	2	Oui	6	0.66	4.79	0.71	0.66	0.62	132	132	0.81
Sigmoid	Sigmoid	4	2	Non	6	0.25	1.43	0.06	0.25	0.10	294	294	0.41

Table 7.1 : Résumé des performances des trois modèles selon la lecture 1

7.3.2 Minimisation des Risques Cliniques

La seconde lecture est axée sur la minimisation des faux négatifs (FN), une priorité dans les contextes cliniques. En médecine, un faux négatif signifie que le modèle ne détecte pas une pathologie — dans notre cas, une tumeur cérébrale — alors qu'elle est présente. C'est donc une erreur de seconde espèce.

En définissant les hypothèses de test suivantes :

$$H_0 : \text{"Le patient n'a pas de tumeur"}$$

$$H_1 : \text{"Le patient a une tumeur"}$$

L'erreur de seconde espèce β est définie comme suit :

$$\beta = \mathcal{P}(\text{Ne pas rejeter } H_0 \mid H_0 \text{ est fausse})$$

Une telle erreur peut entraîner des retards dans le diagnostic et les traitements, avec des conséquences graves pour les patients. À l'inverse, un faux positif, bien que stressant pour le patient, est généralement corrigé après examens complémentaires.

Métriques utilisées : Cette approche met donc l'accent sur le rappel (Recall), qui mesure la proportion de cas positifs correctement identifiés. En maximisant le rappel, le modèle réduit le risque de passer à côté d'un diagnostic critique. Bien que cette stratégie puisse diminuer la précision et augmenter les faux positifs, elle reflète une **priorité éthique et clinique** : ne pas manquer un patient malade. Cependant, minimiser l'erreur de type II risque d'augmenter l'erreur de type I, le nombre de faux positifs (FP). Cela peut réduire la précision globale, mais ce compromis est acceptable en médecine pour des pathologies graves.

Voici la table récapitulative des performances pour nos 3 modèles, selon cette seconde lecture :

Modèle	Activation	Nb couches	Max pooling	Data Aug	Nb d'epochs	Performances globales sans la classe "Pas de tumeur"							
						Acc	Loss	Prec	Recall	F1	FP	FN	AUC
ReLU	ReLU	4	2	Non	6	0.67	4.16	0.72	0.67	0.62	130	130	0.84
Augmented ReLU	ReLU	4	2	Oui	6	0.66	4.79	0.71	0.66	0.62	132	132	0.81
Sigmoid	Sigmoid	4	2	Non	6	0.25	1.43	0.06	0.25	0.10	294	294	0.41

Table 7.2 : Résumé des performances des trois modèles selon la lecture 2

Résumé

Après avoir comparé nos trois modèles sur une base équitable, à nombre de couches, d'epochs, et maxpooling égal, on a dressé un bilan juste de nos modèles, tout en gardant en tête le modèle baseline InceptionV3.

Selon la première lecture, les modèles **ReLU** et **Augmented ReLU** obtiennent des résultats proches en termes d'accuracy (0.67 et 0.66) et de F1-score (0.63 et 0.62), tandis que le modèle **Sigmoid** est nettement moins performant (0.25 d'accuracy et 0.10 de F1-score).

Dans la seconde lecture, qui se concentre sur la réduction des faux négatifs et la maximisation du rappel, les modèles **ReLU** et **Augmented ReLU** conservent des performances similaires avec un rappel de 0.67, mais le modèle **Sigmoid** améliore son rappel (0.25), bien que ses performances globales restent faibles. C'est la lecture adoptée dans le papier "Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation" de [Hajian-Tilaki \(2014\)](#), le rappel reflétant la capacité à identifier les vrais cas malades. Cette approche garantit ainsi un diagnostic plus sûr et aligné sur les priorités éthiques et médicales.

Chapitre 8

Explicabilité et Interprétabilité des modèles

Malgré les progrès du Deep Learning dans les applications médicales, encore peu de travaux intègrent des mécanismes d'explicabilité pour aider les cliniciens à interpréter les résultats des modèles. En plus de limiter l'application de tels modèles dans la vie réelle, cela soulève de véritables questions morales et éthiques. Les réseaux de neurones convolutifs (CNN), grâce à leur capacité à extraire des caractéristiques complexes des images, ont prouvé leur efficacité dans ce domaine. Cependant, leur nature de "boîte noire" rend souvent difficile l'interprétation de leurs décisions, un aspect essentiel en médecine où la compréhension des résultats est cruciale pour la confiance des praticiens.

Définitions

Dans le papier "Explainable Artificial Intelligence (XAI) : Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI" de [Arrieta et al. \(2019\)](#) (page 5), les notions d'explicabilité et d'interprétabilité sont définies de la manière suivante :

L'**explicabilité** est associée à la notion d'explication en tant qu'interface entre les humains et un preneur de décision, qui est à la fois un proxy précis du preneur de décision et compréhensible pour les humains.

L'**interprétabilité** est définie comme la capacité à expliquer ou à fournir la signification en des termes compréhensibles pour un humain.

Démarche et Objectif

Dans cette partie, on s'efforcera de comprendre et **expliquer** les résultats de nos modèles, en particulier le modèle ReLU sans augmentation de données. Nous allons donc explorer les méthodes d'explicabilité pour analyser et interpréter les décisions d'un modèle CNN appliqué à des images IRM de cerveaux, classées en fonction de la présence ou de l'absence de tumeurs.

Nous tenterons de répondre aux questions suivantes :

- Quelles parties de l'image influencent le plus les prédictions du modèle ?
- Le modèle est-il biaisé par certaines zones non pertinentes ?

Package Xplique

[Xplique](#) est une bibliothèque d'explicabilité des modèles d'apprentissage profond. Elle fournit plusieurs outils, appelés explainers, permettant de comprendre comment un modèle prend ses décisions. Les explainers que nous avons choisis pour cette analyse (`IntegratedGradients`, `SmoothGrad`,

GradCAM et Guided Backpropagation) reposent tous sur les gradients et la backpropagation. Ces techniques permettent de déterminer quels pixels de l'image influencent le plus la prédiction du modèle.

Application du package à nos données

Une fois les données et le modèle importés, nous pouvons appliquer les fonctions du package. Afin d'éviter le phénomène de saturation des activations, qui nuisent à certaines techniques d'explicabilité (GuidedBackprop, IntegratedGradients, SmoothGrad ou GradCAM dans notre cas), nous allons travailler directement avec les sorties brutes de notre modèle avant activation, appelées **logits**.

On peut ensuite sélectionner une liste d'explainers :

```
1 explainers = [
2     GuidedBackprop(model),
3     IntegratedGradients(model, steps=80, batch_size=batch_size),
4     SmoothGrad(model, nb_samples=80, batch_size=batch_size),
5     GradCAM(model)
6 ]
```

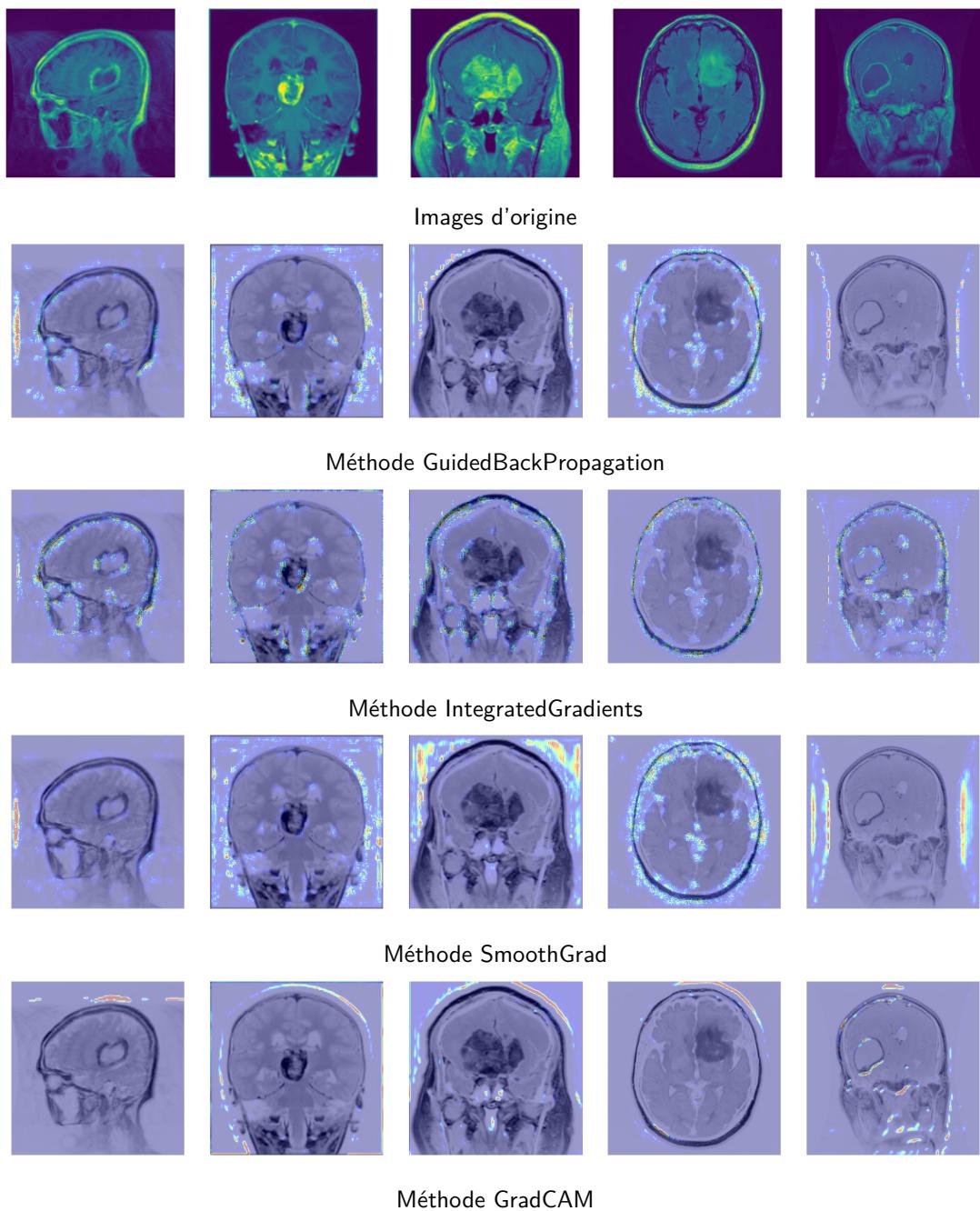
Listing 8.1 – Liste des explainers sélectionnés

Voici les méthodes utilisées dans ce projet, et l'interprétation qu'on a pu tirer des images (voir page suivante) :

- **Integrated Gradients** : Les cartes produites par Integrated Gradients montrent un détourage précis du cerveau sans apparition de rectangle parasite. La méthode identifie clairement les contours externes du cerveau, mais elle attribue peu d'importance aux structures internes, comme les tumeurs. Cependant, lorsque les pixels internes sont supprimés, les performances du modèle diminuent, ce qui indique qu'ils ont tout de même une certaine influence. Integrated Gradients semble ainsi bien capturer les contours du cerveau, mais reflète potentiellement un biais où les caractéristiques internes sont sous-évaluées.
- **SmoothGrad et Guided Backpropagation** : SmoothGrad et Guided Backpropagation offrent des perspectives similaires sur les activations du modèle. Les deux méthodes mettent en évidence un détourage précis du cerveau, souvent accompagné d'une forme rectangulaire récurrente. Cette forme semble jouer un rôle important dans les prédictions du modèle, ce qui suggère un possible biais systématique lié à l'acquisition des IRM ou à des caractéristiques invisibles à l'œil nu. SmoothGrad, en produisant des cartes d'activations lisses et raffinées, met en avant les contours périphériques, tandis que Guided Backpropagation, avec ses détails texturaux plus fins, montre que ces bords dominent également les activations.
- **Grad-CAM** : GradCAM met en évidence une forte activation des pixels entourant le cerveau, suggérant que les contours périphériques jouent un rôle clé dans les décisions du modèle. Cependant, les structures internes du cerveau sont également bien colorées, ce qui montre qu'elles influencent significativement les prédictions.

Étant donné que les images originales sont au format RGB, leur conversion en niveaux de gris lors du prétraitement, combinée à l'utilisation de la bibliothèque matplotlib pour leur affichage, produit un rendu visuellement "coloré".

En générant des attributions pour chaque méthode listées ci-dessus, on peut afficher les images expliquées, qu'on compare avec les images d'origine (voir page suivante).

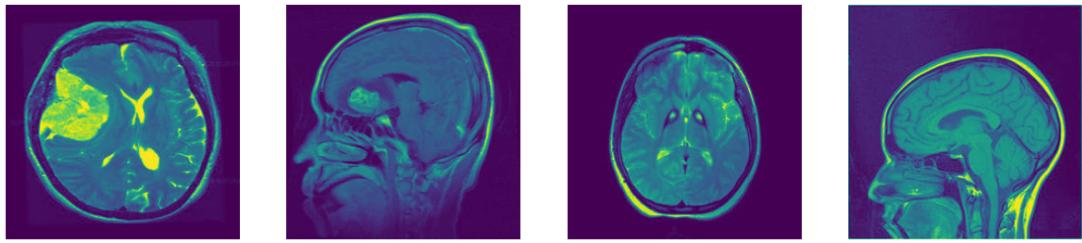


Hypothèses sur les Résultats Observés

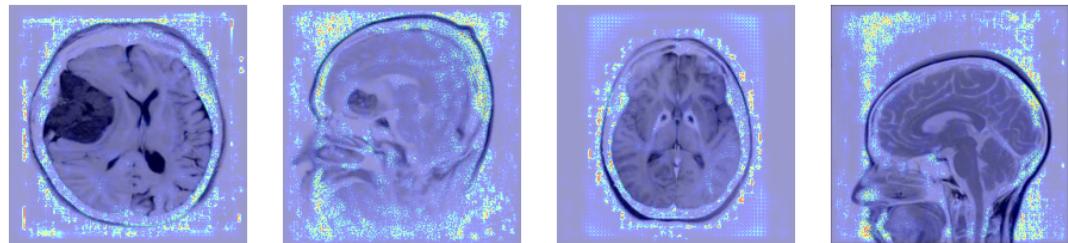
- **Impact des outils d'acquisition** : Le rectangle observé dans certaines méthodes pourrait être lié à des artefacts techniques propres aux appareils d'IRM.
- **Biais dans les données** : Le modèle semble accorder trop d'importance aux contours du cerveau et moins aux anomalies internes.
- **Variabilité entre appareils** : Les différences d'appareils ou de protocoles pourraient expliquer l'apparition de caractéristiques comme les rectangles.

Exploration d'images correctement prédictées

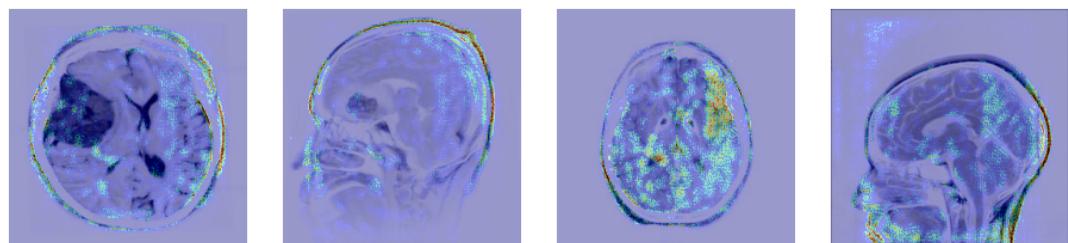
Voyons maintenant comment le modèle se comporte sur des images correctement prédictées. Ces résultats permettront d'explorer les limites atteignables en terme d'explicabilité avec le package.



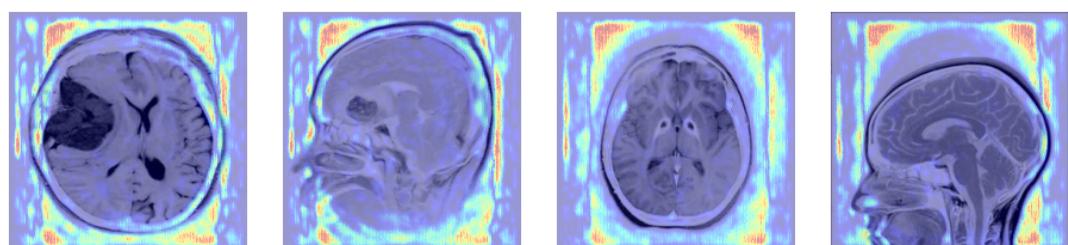
Images d'origine correctement prédites



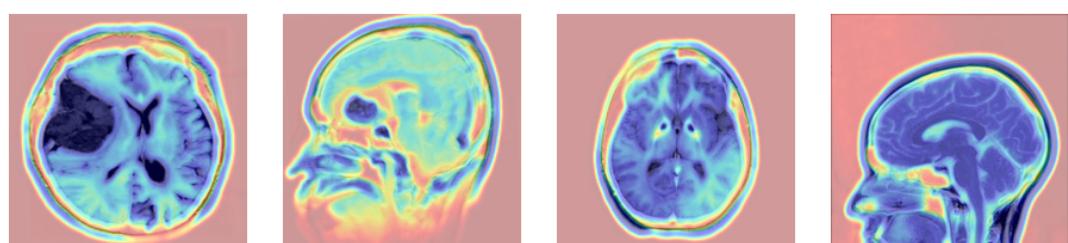
Méthode GuidedBackPropagation sur des images correctement prédites



Méthode IntegratedGradients sur des images correctement prédites



Méthode SmoothGrad sur des images correctement prédites



Méthode GradCAM sur des images correctement prédites

Conclusion

L'analyse des explainers fournis par Xplique révèle que le modèle présente un comportement cohérent dans la mise en évidence des contours cérébraux. Une exploration complémentaire pourrait inclure une normalisation des données ou une analyse des différences entre appareils d'acquisition.

Chapitre 9

Conclusion

Dans ce projet, nous avons développé un modèle de détection de tumeurs cérébrales à partir d'IRM en utilisant des réseaux de neurones convolutifs (CNN). Notre modèle a permis de classer les images en quatre catégories : les trois types de tumeurs identifiés et l'absence de tumeur. Ce travail a été marqué par plusieurs défis et limitations.

L'une des principales contraintes a été la limitation des ressources de calcul, ce qui nous a conduits à restreindre l'entraînement à seulement six époques. Cela a également influencé notre capacité à explorer un éventail plus large de modèles ou à tester des outils d'explicabilité avancés, comme certains explainers nécessitant une puissance de calcul importante. Ces limitations techniques nous ont empêchés d'optimiser pleinement les performances du modèle et de maximiser sa transparence.

Nous avons cependant veillé à intégrer la spécificité du contexte médical dans notre approche. En raison des enjeux critiques liés à la santé des patients, nous avons conçu notre évaluation pour minimiser les faux négatifs. Cela signifie que le modèle priviliege la détection d'une tumeur même en l'absence de celle-ci, afin d'éviter des diagnostics manqués, qui auraient des conséquences graves.

Ce projet a également été une excellente opportunité d'apprentissage. Nous avons approfondi nos connaissances sur les CNN au-delà du cadre du cours en explorant les subtilités de leur fonctionnement. Avant de nous attaquer à ce projet complexe, nous avons pris le temps de tester des modèles sur des ensembles de données plus simples, ce qui nous a permis de progresser étape par étape avec une meilleure maîtrise des concepts fondamentaux.

Nous avons respecté les bonnes pratiques en informatique, en adoptant une structuration rigoureuse du code, en documentant nos travaux et en mettant en œuvre des processus collaboratifs efficaces. Cependant, certaines leçons importantes ont émergé en cours de projet. Par exemple, nous avons réalisé tardivement qu'un traitement des images en RGB aurait pu améliorer les performances, ce qui constitue une piste d'amélioration. De même, avec des ressources accrues, nous aurions souhaité augmenter le nombre d'époques pour affiner l'apprentissage du modèle.

En conclusion, ce projet a représenté un défi enrichissant, tant sur le plan technique que méthodologique. Bien que perfectible, il nous a permis d'acquérir des compétences solides en apprentissage profond, de mieux comprendre les enjeux médicaux, et d'identifier des axes d'amélioration clairs pour de futurs travaux.

Chapitre 10

Discussion

10.1 Limitations

Limites computationnelles

Les modèles CNN (Convolutional Neural Networks) étant des réseaux de neurones particulièrement lourds, leur temps d'entraînement augmente significativement avec la taille des données et la profondeur du réseau. Bien que l'augmentation ne soit pas strictement exponentielle, elle est généralement superlinéaire, en raison de la multiplication des opérations nécessaires au calcul des convolutions, des gradients et de l'optimisation des paramètres. Dans notre cas, nous avons été contraints de limiter l'entraînement à 6 epochs pour garantir une comparaison équitable entre les trois modèles testés, compte tenu de nos ressources computationnelles limitées.

Taille et qualité des données

La qualité et la quantité des données d'entraînement influencent fortement les performances des modèles CNN. Un jeu de données limité ou déséquilibré peut conduire à un surapprentissage (overfitting) ou à des biais dans les prédictions. Dans ce projet, nous avons utilisé un ensemble de données fixe et n'avons pas pu explorer l'impact de l'augmentation significative des données sur les performances des modèles.

Problèmes de généralisation

Les modèles CNN, bien qu'excellents sur les données d'entraînement, peuvent parfois manquer de capacité de généralisation, notamment lorsqu'ils rencontrent des données provenant d'une distribution différente ou de nouvelles variations. Cette limitation est particulièrement prononcée dans des scénarios où les données augmentées n'englobent pas suffisamment de diversité.

Choix des hyperparamètres

Le choix des hyperparamètres, comme la taille des couches convolutives, le nombre de filtres, les taux d'apprentissage ou les fonctions d'activation, influence directement les performances des modèles. Dans ce projet, les hyperparamètres ont été fixés de manière uniforme pour comparer les modèles, mais cela a pu empêcher une optimisation fine spécifique à chaque architecture.

Limitations liées au benchmark

La limitation à 6 epochs, bien qu'indispensable pour garantir une équité dans la comparaison des modèles, a restreint la capacité des réseaux à converger pleinement. Par conséquent, les performances obtenues peuvent ne pas refléter le plein potentiel de chaque architecture dans des conditions d'entraînement prolongées.

Biais introduits par les données

Enfin, les biais présents dans les données d'entraînement peuvent être amplifiés par les modèles CNN. Dans notre projet, nous n'avons pas pu effectuer une analyse exhaustive des biais potentiels dans le jeu de données utilisé, ce qui constitue une limitation à prendre en compte dans l'interprétation des résultats.

10.2 Axes d'amélioration

Fine-tuning du modèle ReLU

Dans le cadre du développement de l'application Streamlit, permettant une utilisation interactive du modèle, nous avons fait un finetunning du meilleur modèle en ajoutant 10 epochs supplémentaires (pour un total de 16). De cette manière, nous avons obtenu un modèle plus performant, atteignant environ 71% d'accuracy. Ces performances s'approchent de celles d'InceptionV3. (Voir le modèle `modele_brain_tumor_20241214_174158_finetuning.h5` dans [ce fichier .json](#)).

Hypothèse Grayscale vs RGB

Dans ce projet, les données IRM utilisées étaient déjà en format RGB malgré leur apparence en noir et blanc. Étant donnée la nature monochrome des IRM et la complexité élevée des données RGB, il semblait pertinent de convertir les images en grayscale pour simplifier le modèle et réduire la charge computationnelle.

Cependant par la suite, lors de l'entraînement d'un même modèle CNN, deux versions ont été comparées : l'une avec les données RGB d'origine et l'autre avec des images converties en grayscale. Les performances obtenues ont montré une précision de 74% (`model_20241215_142118`) avec le format RGB contre 71% (`modele_brain_tumor_20241214_174158_finetuning.h5`) avec le format grayscale. Les résultats des deux modèles sont consultables dans [ce fichier .json](#))

Cette différence peut sûrement s'expliquer par les facteurs suivants : premièrement, la réduction de dimension cause généralement une perte d'information. Les canaux RGB, bien qu'observés en noir et blanc, ne sont pas toujours identiques, ce qui peut introduire des variations entre les canaux, exploitables par le modèle. Enfin, la redondance entre les canaux RGB peut agir comme une forme implicite de régularisation.

Même si les images IRM n'ont pas de variations intentionnelles entre les canaux, les légères discordances naturelles dans le format RGB peuvent jouer un rôle similaire à celui des techniques d'augmentation par PCA dans ImageNet effectués dans le papier "ImageNet Classification with Deep Convolutional Neural Networks" de Krizhevsky et al. (2012). Les différences fournissent au modèle des indices supplémentaires ou renforcent sa capacité à généraliser, en le rendant moins dépendant de l'uniformité des pixels. Ces résultats montrent que, bien que la conversion en grayscale semblait justifiée pour réduire la complexité, le format RGB a permis de capturer des informations supplémentaires ou d'exploiter des mécanismes architecturaux adaptés, aboutissant à de meilleures performances.

Considérations énergétiques

Enfin un dernier axe d'amélioration et pas des moindres, concerne l'empreinte énergétique de ces algorithmes. En effet, ces modèles CNN nécessitant une puissance de calcul très élevée, à tel point que nous avons dû nous restreindre à un nombre limité d'epoch. Cet aspect est une considération importante, surtout dans un contexte de recherche visant à adopter des solutions durables et respectueuses de l'environnement.

Bibliographie

- Arrieta, A. B., Díaz-Rodríguez, N., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R. and Herrera, F. (2019), 'Explainable artificial intelligence (xai) : Concepts, taxonomies, opportunities and challenges toward responsible ai'.
- Bhuvaji, S., Kadam, A., Bhumkar, P., Dedge, S. and Kanchan, S. (2020), 'Brain tumor classification (mri)'.
- Bossé, Y., Mathieu, P. and Pibarot, P. (2017), 'Genetic study of calcific aortic valve disease : A disease with developmental origins', *Journal of Cardiology* **69**(5), 417–424.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009), 'Imagenet : A large-scale hierarchical image database', *Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 248–255.
- Hajian-Tilaki, K. (2014), 'Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation', *Journal of Biomedical Informatics* **48**, 112–115.
- He, K., Zhang, X., Ren, S. and Sun, J. (2016), 'Deep residual learning for image recognition', *Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 770–778.
- Horsley, V. (1906), 'Discussion on the treatment of cerebral tumors', *British Medical Journal* **2**, 411–423.
- Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in 'Advances in Neural Information Processing Systems (NIPS)', Vol. 25, Curran Associates, Inc., pp. 1097–1105.
- LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.
- Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R. et al. (2015), 'The multimodal brain tumor image segmentation benchmark (brats)', *IEEE transactions on medical imaging* **34**(10), 1993–2024.
- Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M. and Salem, A.-B. M. (2018), 'Classification of brain tumor types by a deep convolutional neural network using mri images', *Egyptian Informatics Journal* **19**(3), 179–189.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2014), 'Going deeper with convolutions'.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016), 'Rethinking the inception architecture for computer vision', *Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 2818–2826.
- Tustison, N. J., Avants, B. B., Cook, P. A., Yuan, Y., Gee, J. C. and Song, G. (2010), 'N4ITK : Nick's N3 ITK implementation for MRI bias field correction', *IEEE Transactions on Medical Imaging* **29**(6), 1310–1320.

Annexe A : Bonnes pratiques et Arborescence du projet

Dans un soucis de rendre nos modèles les plus reproductibles possibles, nous avons tenu à suivre quelques bonnes pratiques en informatique, lorsqu'il s'agit de développements de modèles de Deep Learning. Cela facilitera l'onboarding de nouveaux contributeurs et la maintenance du projet sur le long terme.

Bonnes pratiques en informatique

Dans le cadre de notre projet de détection de tumeurs cérébrales, nous avons adopté plusieurs bonnes pratiques en informatique afin d'assurer la clarté, la reproductibilité et la facilité de collaboration. Ces pratiques incluent :

Organisation de l'arborescence du projet

Nous avons structuré le projet en suivant une arborescence standard (consultable p.38) pour les projets de data science. Les principaux répertoires incluent :

- data : pour contenir les données avec des sous-dossiers comme raw (données brutes) et processed (données prétraitées),
- Data_visualisation : pour les scripts et résultats de visualisation,
- results : contenant les sorties comme les prédictions correctes, les fichiers d'explicabilité et les rapports de performance (e.g., metrics.html),
- src : regroupant les sous-dossiers models (scripts liés aux modèles comme CNN_app.py, CNN_train.py, etc.) et preprocessing (pour les scripts de prétraitement),
- venv : pour l'environnement virtuel et les dépendances associées.

Utilisation d'un fichier de configuration (config.yaml)

Ce fichier centralise les paramètres du projet, comme les chemins des données, certains hyperparamètres des modèles et les configurations d'entraînement. Cela permet de garantir la reproductibilité et de faciliter les modifications : il suffit de changer le fichier YAML sans altérer le code.

Script de lancement (lancez_moi.sh)

Nous avons écrit un script Bash pour simplifier la prise en main du projet. Ce script automatise les étapes principales, comme l'installation des dépendances, la configuration de l'environnement virtuel et l'exécution du processus principal.

Environnement virtuel

Nous avons travaillé dans un environnement virtuel (venv) pour isoler les dépendances du projet et éviter les conflits avec d'autres projets ou configurations systèmes. Cela garantit un environnement propre et reproductible.

Gestion de version avec Git et GitHub

Nous avons utilisé GitHub pour versionner le projet, ce qui permet de suivre l'évolution du code et de collaborer efficacement. Voici quelques éléments-clés :

- **Création de branches** : chaque fonctionnalité ou expérimentation a été développée dans une branche séparée, facilitant ainsi la gestion des versions et la résolution de conflits.
- **Commits réguliers et descriptifs** : chaque modification a été sauvegardée avec des messages au possible explicites pour tracer l'historique des changements.
- **Utilisation de pull requests** : pour revoir et valider les modifications avant leur fusion dans la branche principale.

Bonnes pratiques pour les projets de traitement d'images

Avant de traiter directement les images IRM, nous avons commencé par utiliser les données CIFAR-10, qui contiennent 10 classes (comme camion, oiseau, etc.). Cela nous a permis de former de petits modèles de reconnaissance de classes en un temps réduit (évalués à moins de 15 minutes par itération) pour tester plusieurs combinaisons d'hyperparamètres. Cette phase a été essentielle pour développer une intuition sur les modèles et leurs comportements.

Par ailleurs, nous nous sommes inspirés d'autres projets similaires disponibles sur Internet, comme un projet de classification de types de riz (à consulter [ici](#)), qui a été une source d'inspiration précieuse au début de notre travail. Ces pratiques ont permis de structurer le projet de manière efficace et de garantir sa maintenance à long terme tout en facilitant la collaboration avec d'autres membres éventuels.

Arborescence du projet

Voici l'arborescence du projet :

```
Brain_tumor/
├── README.md
├── requirements.txt
├── config.yml
└── Data_visualisation/
    └── Data.visualization.html
src/
├── preprocessing/
    ├── module_for_preprocessing.py (module avec les fonctions codées pour le
        preprocessing)
    └── preprocessing.py (création des folders avec les images prétraitées)
└── models/
    ├── CNN_train.py (pour créer un modèle)
    ├── CNN_finetunning.py (pour ajouter des epochs si on le souhaite)
    ├── CNN_.py (sauvegarde des modèles)
    └── sauvegarde_models/
        ...sauvegardés en externe sur nos machines (trop lourds pour GitHub)
        └── streamlit.py
results/
├── module_for_preprocessing_metrics.py
├── metrics.ipynb
├── metrics.html
└── explicability.py
    └── explicability.html
lancez-moi.sh (commande qui : crée un environnement virtuel comprenant les
librairies nécessaires pour la visualisation des résultats et l'active, ouvre
les fichiers data_visualisation.html, metrics.html et explicability.html sur
votre page web, et lance l'application Streamlit)
```