SYS843 - Étude expérimentale

Comparaison de modèles pour la classification de tumeurs cérébrales

Le génie pour l'industrie

Soumis par :
M. Ribeiro Lucas
(RIBL05279901)

A destination de :
Prof. Eric Granger

Session Automne 2022

20 Décembre 2022

Table des matières

Table des figures

1 Mise en situation

1.1 Domaine d'application

Actuellement, le domaine médical possède de nombreux outils afin d'analyser le corps humain. L'imagerie à résonnance magnétique (IRM) en fait donc partie et se voit d'une grande efficacité afin de visualiser les tissus mous du cerveau. Les IRM permettent d'obtenir des vues du cerveau en 2D ou en 3D, ces images peuvent être de deux contrastes différents T1 ou T2. Les temps de répétition et d'écho sont également ajustables. La complexité de ces paramètres, peuvent alors refléter la difficulté quand à l'analyse de ces données. En effet, lors de la prise d'image IRM, il est important de prendre en compte ces différents paramètres, ils peuvent amener à des résultats qui peuvent différer même pour une seule et même personne. Il est alors judicieux de se questionner sur la qualité des données. Elles peuvent être insuffisantes, ou nécessitent différents pré-traitements avant d'être exploitées. C'est donc à partir des images IRM que les tumeurs cérébrales sont aujourd'hui détectées et localisées. Ces dernières peuvent être à différentes phases ou de différents types, leurs formes et leurs tailles diffèrent grandement. Il vient par la suite que cette détection ne peut être réalisée sans la présence d'un spécialiste, le radiologue. Cependant, son analyse va généralement être influencé par son expérience accumulé au fil des années. Ce spécialiste peut donc avoir certaines lacunes, lorsqu'il s'agit de maladie qu'il ne rencontre pas au cours de sa carrière. La création de nouveaux outils permet donc d'améliorer ses diagnostics, mais n'ont pas pour objectif de le remplacer.

1.2 Problématique abordée

La problématique abordée cache en réalité plusieurs sous problématiques. Certaines proviennent du domaine d'application et d'autres de la mise en place de classificateurs.

La mise en place de classificateurs, demande alors une base de données assez importante et spécifique à un certains types de tâches données. Actuellement, dans le domaine de classification de tumeurs du cerveau, les classificateurs existants sont de deux types (Cheng et al., 2015) :

— Deux classes, normale et anormale.

— Plusieurs classes malignes (glioma, meningioma, etc.).

Il est également important de savoir, que les classificateurs afin d'être plus performants ne traitent pas les données brutes directement. Il y a une partie d'extraction des données qui peut se faire via des algorithmes, ou bien des réseaux de neurones artificiels (RNA), puis par la suite de traiter ces données extraites dans un classificateur. Par ailleurs, la variabilité des données brutes peuvent grandement altérer le résultat. Un modèle d'apprentissage peut être très performant pour des réglages bien spécifiques, mais ne plus fonctionner lorsque ces réglages viennent à changer. Cela peut entraîner des différences qui ne sont pas perceptibles par l'être humain sur une image, mais qui impacte grandement le modèle lors de l'analyse de l'image.

Cela permet donc de se questionner sur l'existence d'un modèle, le plus adapté, afin de classer de tumeurs cérébrales. Dans un premier lieu, il est important de définir les données qui seront traitées, ainsi que leurs cohérences avec la tâche à réaliser. Et dans un

second lieu, de déterminer le modèle qui répond le mieux à cette tâche de classification en comparant différents modèles.

1.3 Objectifs du projet

L'objectif de ce projet est de comprendre le fonctionnement des outils de classification de tumeurs du cerveau déjà présents dans le but de recherche et de comparer trois approches parmis celles-ci. C'est à dire, de comparer des classificateurs dans la classification du types de tumeurs déjà présentes. Il s'agira de l'utilisation d'un CNN avec augmentation de données (Sadoon and Ali, 2021). Et puis, on verra également l'utilisation d'un modèle dejà existant (Szegedy et al., 2015) et l'utilisation d'un modèle à apprentissage par transfert combiné d'un SVM (Rehman et al., 2020).

Ces modèles permettront d'appercevoir différentes approches afin de classer des données provenant d'images IRM. Les différences qu'ils possèdent, ainsi que leurs forces et faiblesses pourraient laisser penser, qu'un modèle combinant leurs forces et atténuant leurs faiblesses pourraient permettre un gain en robustesse très important.

1.4 Structure du document

Dans cette étude expérimentale, nous allons faire un rappel des techniques utilisées à savoir l'utilisation d'un CNN sans nom, de InceptionV3 sans pré-entraînement et de

InceptionV3 freeze couplé à un SVM. Puis nous verrons la méthodologie appliquée, c'est-à-dire : le protocole d'évaluation, la description de la base de données et les mesures de performances. Ensuite, les résultats seront présentés et interprétés. Enfin, nous conclurons sur des recommandations pour la suite du projet, et faire un court débrief du projet.

2 Sommaire des techniques utilisées

Les approches que je vais à présent décrire, ont toutes été mise en place à partir de la même base données. Cela me permettra de comparer ces méthodes entre elles, de façon pertinente plus tard.

Cependant, il faut dans un premier temps décrire le formalisme mathématique d'un CNN.

2.1 Formalisme mathématique d'un CNN

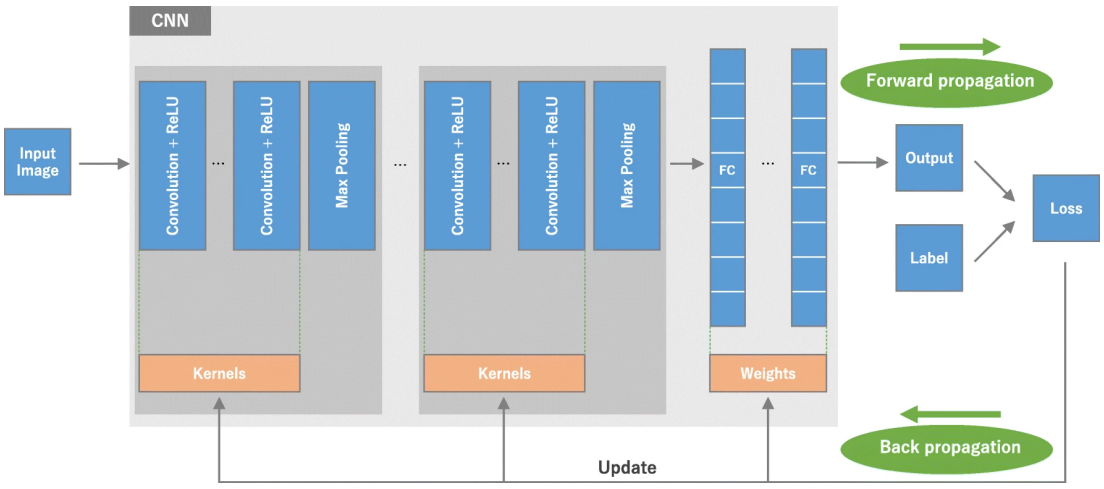

Figure 1 – Architecture classique d'un CNN

La Figure 1 présente une architecture possédant plusieurs noyaux de couches cachées convolutionnelles avec la fonction d'activation ReLU (1) qui est une des fonctions les plus utilisées en imagerie. Les couches de sorties sont des couches complètement connectées (FC).

$$f(x) = max(0, x) \tag{1}$$

Couche de convolution (Pashaei et al., 2018)

La couche de convolution est connectée par des neurones aux filtres, où la taille des neurones est spécifiée sur l'image d'entrée. Chaque filtre possède une matrice de poids. Le nombre de poids dans un filtre peut donc être calculé selon (2) :

$$K \times M \times N \tag{2}$$

où M, N, K sont respectivement la hauteur, la largeur et le nombre de canaux du filtre d'entrée.

Le nombre de paramètres d'une couche de convolution se calcule de la manière suivante (3) :

$$(K \times M \times N + 1) \times nombredefiltres \tag{3}$$

où le *+1* correspond à l'ajout du biais, K,M,N n'ayant pas changé.

Enfin pour connaître la taille de la sortie, on peut se réferer au calcul suivant (4) :

$$\frac{(inputsize - filtersize + 2 \times padding)}{stride} + 1 \tag{4}$$

où stride correspond au pas du déplacement du filtre et padding permet d'assurer la cohésion du filtre avec l'image d'entrée.

Couche de maxpooling

Le maxpooling est une opération de pooling qui calcule la valeur maximale des zones d'une carte de caractéristiques et l'utilise pour créer une nouvelle carte de caractéristiques sous-

échantillonnée. Ici, il s'agira généralement des sorties de couches de convolutions décrites plutôt.

Couche de batch normalization

Cette couche permet de normaliser l'entrée x_i en calculant dans un premier temps la moyenne μ_B et la variance σ_B^2 à travers un mini-batch et au travers de chaque canaux d'entrée. Ensuite, elle calcule la fonction d'activation normalisé (5) :

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 - \epsilon}} \tag{5}$$

où ϵ permet d'assurer la stabilité lorsque la variance d'un mini-batch devient trop proche de zéro.

Couche Softmax

Cette couche permet de calculer une prédiction de classe. Cela consiste à créer une distribution probabilistique de chaque sortie à partir d'un vecteur de K nombres réels. Son calcul (6) est le suivant :

$$\sigma(\mathbf{y})_i = \frac{\exp y_i}{\sum_{j=1}^{K} \exp y_j} \tag{6}$$

2.2 CNN sans nom avec augmentation de données

Augmentation de données

La première étape de ce modèle est d'appliquer de l'augmentation de données ainsi qu'un pré-process de ces dernières. Pour ce faire, il existe de multiples méthodes dans le milieu médical afin de procéder à de l'augmentation de données, notament dans le sujet de l'imagerie.

Les techniques basiques d'augmentation de données sont les suivantes :

— **Transformation géométriques :** le redimensionnement, la translation, la rotation, la réflection ou encore le cisaillement. Les transformations géométriques sont souvent les plus utilisées afin de procéder à de l'augmentation de données. La raison provient de la facilité d'application de ces dernières.

— **Cropping (rognage) :** cette technique consiste à extraire un parcelle de l'image d'origine afin de l'ajouter dans la base de données.

— **Occlusion :** cette technique retire des parcelles de l'image et conserve son état.

— **Operation d'intensité :** cette opération revient à modifier les contrastes de l'image, de jouer sur sa luminosité. Cette méthode est également très courante.

— **Injection de bruit :** une technique populaire qui revient à ajouter du bruit Gaussien, ou encore du bruit poivre et sel.

— **Filtre :** cette méthode utilise des filtres de convolutions afin de lisser, flouter, ou ajouter de la netteté sur les images. Cela peut également permettre de faire de la détection de bordure.

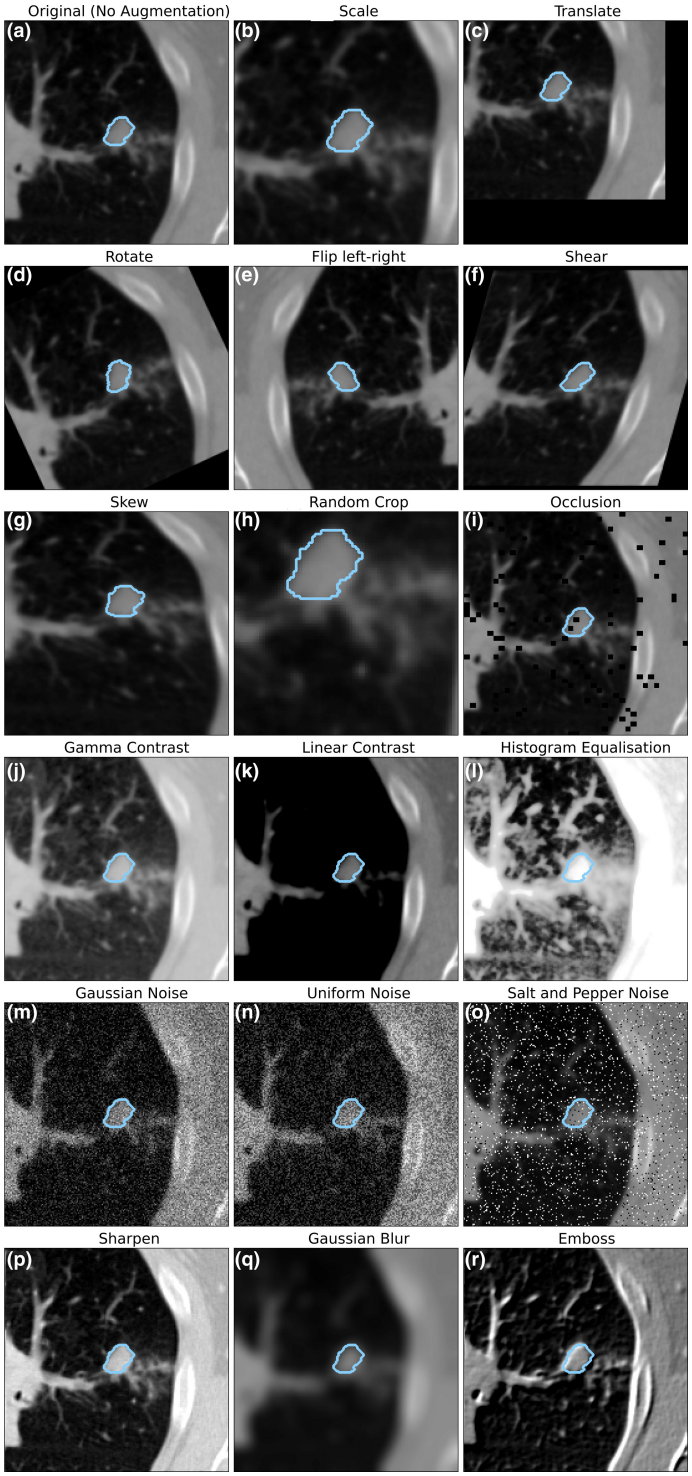

Figure 2 – Exemples de techniques d'augmentation de données appliqué sur des images médicales (Chlap et al., 2021).

Sur la figure (Fig. 2), on peut observer les différents effets de techniques d'augmentation

de données enoncées précédemment.

Dans mon cas, j'appliquerai uniquement le flip horizontal et vertical sur la totalité de ma base de données. Cette limitation provient de la limite de mémoire vive de mon environnement d'exécution.

CNN

Après avoir augmenter les données, la méthode décrite met en place un CNN avec de nombreuses couches de convolution puis effectue des tests en changeant certains paramètres, voire hyperparamètres afin d'obtenir le modèle souhaité. Le CNN est un réseau neuronal utilisé pour classer les images et a montré de bonnes performances dans la catégorisation de différentes tâches d'apprentissage supervisé (Sadoon and Ali, 2021).

Le modèle ainsi présenté comporte 28 couches, dont la première est la couche d'entrée qui prends des images 128x128x3 après la sortie du pré-process et de l'augmentation de données.

Les couches suivantes sont respectivement une couche de convolution, une couche softmax et un maxpooling, que l'on met six fois d'affilée. Il y a également la présence de cinq couches de dropout au sein du CNN afin d'éviter la suradaptation. Les trois couches finales sont successivement une couche FC, une couche softmax et enfin la couche de classification. Il est également précisé qu'il y a un ajout d'une couche de batch normalisation après la première couche de convolution.

Description des couches de convolutions dans l'ordre respectif (Sadoon and Ali, 2021) :

— Nombres de filtres : 64, 64, 96, 96, 128 et 128.

— Tailles des kernel : 7x7, 9x9, 9x9, 9x9, 11x11 et 11x11.

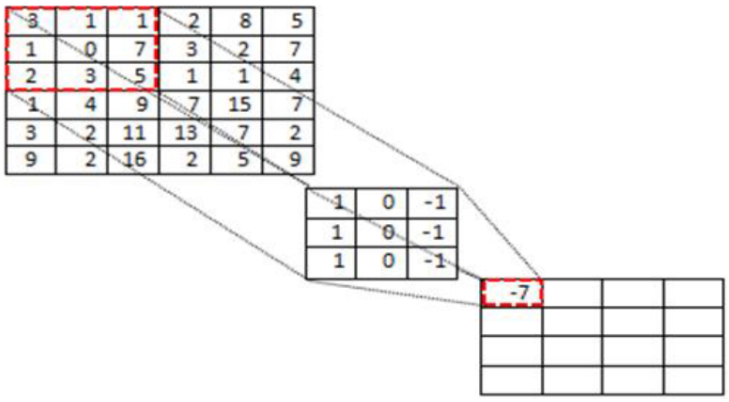

Figure 3 – Exemple d'application d'une couche de convolution(Sadoon and Ali, 2021).

— Taille du padding : 0, 1, 1, 1, 1 et 1.

La sortie des couches de convolutions sont alors de nouvelles images, correspondant à des cartes de caractéristiques. La Figure 3 montre un exemple d'opération de convolution avec un kernel 3x3.

La fonction ReLU (1) est la fonction d'activation utilisé en sortie de chacunes des couches de convolution.

Afin d'éviter la présence de suradaptation, le modèle préconise l'utilisation de dropout. Ses recommandations sont respectivement pour les cinq dropout appliqué : 10%, 10%, 20%, 20% et 20%. Un exemple de dropout est représente sur la Figure 4.

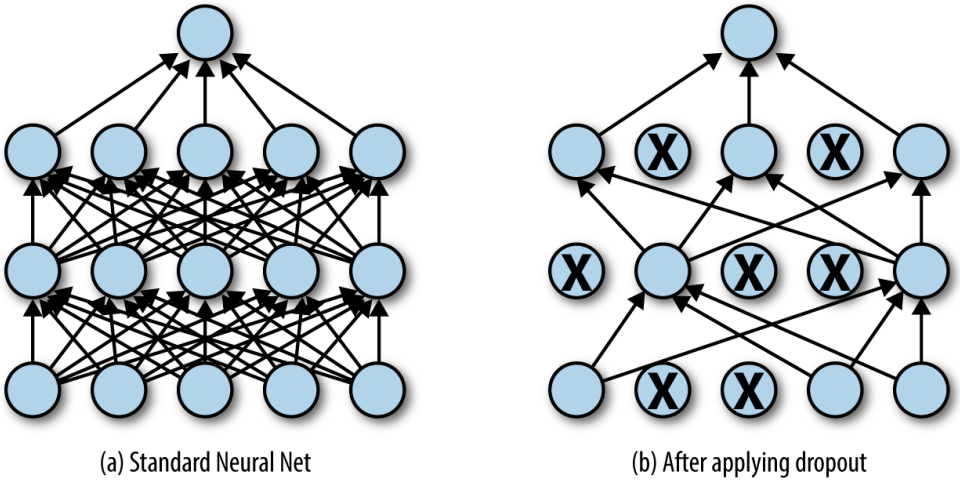

Figure 4 – Exemple de dropout (oreilly.com)

La couche FC utilisé permet avant d'appliquer le Softmax de linéariser la sortie sous forme de vecteur. Enfin, Sadoon and Ali (2021) utilise la fonction de cross-entropy loss afin de calculer l'erreur de prédiction du classificateur. L'équation (7) se présente sous la forme suivante :

$$E = \sum_{i=1}^{N} (-d_i ln(y_i) - (1 - d_i) ln(1 - y_i)) + \lambda \frac{1}{2} \|w\|^2 \tag{7}$$

où y_i est la sortie du classificateur, d_i est la réponse souhaitée, λ est le taux d'apprentissage, et N le nombre de classe.

Le modèle utilise également une méthode d'optimisation afin de réduire l'erreur : Stochastic Gradient Descend with Momentum (SGDM). Il est tout de même possible d'utiliser l'algorithme Adam pour l'optimisation.

Dans mon cas, j'ai été limité en terme de capacité du processeur graphique (GPU). De ce fait, je n'ai pas pu implémenter les couches de convolutions telles que décrites précédemment. J'ai donc décidé de les modifier.

Description modifiée des couches de convolutions dans l'ordre respectif :

— Nombres de filtres : 32, 32, 32, 32, 64 et 64.

— Tailles des kernel : 7x7, 9x9, 9x9, 9x9, 11x11 et 11x11.

— Taille du padding : 0, 1, 1, 1, 1 et 1.

Les autres éléments décrits précedemment sont eux conserver.

2.3 InceptionV3 + SVM

InceptionV3

InceptionV3 est un modèle de réseau de neurones convolutif (CNN) utilisé pour la reconnaissance d'images. Il a été présenté par Google en 2015 dans l'article de Szegedy et al. (2015).

InceptionV3 est un modèle très performant et largement utilisé dans les domaines de la vision par ordinateur et de l'apprentissage automatique en général. Il a été entraîné sur un grand nombre d'images et il est capable de reconnaître avec une grande précision différents objets, animaux, paysages, etc.

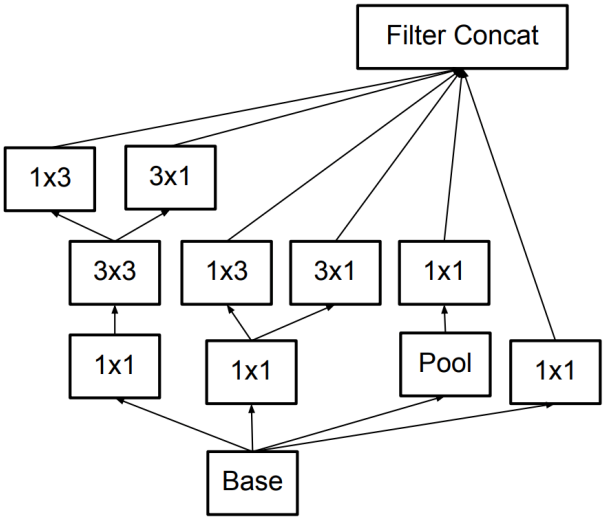

Figure 5 – Module InceptionV3 (Szegedy et al., 2015).

Le modèle InceptionV3 est basé sur le concept de "module inception", qui consiste en une combinaison de filtres de convolution de tailles différentes, de pooling et de couches de normalisation, le tout connecté en parallèle. Cette architecture permet au modèle de capturer des caractéristiques à différentes échelles, ce qui lui permet de mieux gérer la

variabilité des images et d'obtenir de meilleures performances.

En plus de sa performance élevée, un autre avantage de InceptionV3 est qu'il est relativement rapide à entraîner et à utiliser en pratique. C'est la structure du module inception qui permet de réduire le nombre de paramètres du modèle et le temps d'entraînement.

InceptionV3 est disponible dans plusieurs frameworks de deep learning populaires, tels que TensorFlow et PyTorch. Ce modèle est souvent utilisé comme base pour des tâches de reconnaissance d'images de haute qualité.

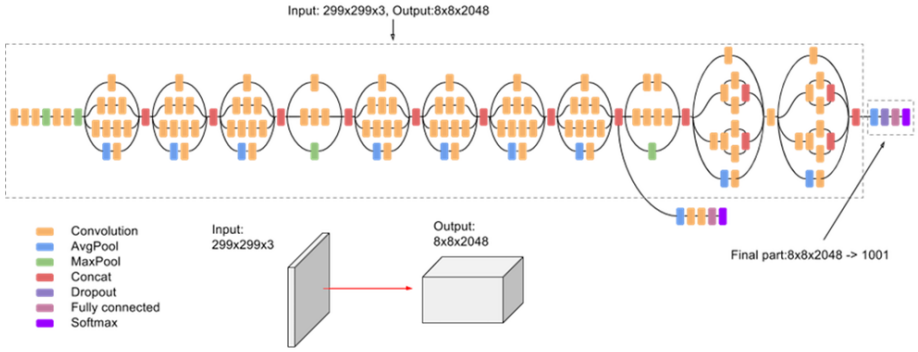

Figure 6 – InceptionV3 Architecture (Google, na).

SVM

Un Support Vector Machines (SVM) est un classificateur basé sur des méthodes de supervision afin de faire de la classificiation, de la régression ou encore de la détection de bordure. Sa particularité est la possibilité de changer sa fonction noyau, il en existe une multitude : linéaire, quadratique, polynomial, Radial Basis Function (RBF), sigmoid (Srinivasa Reddy and Chenna Reddy, 2021).

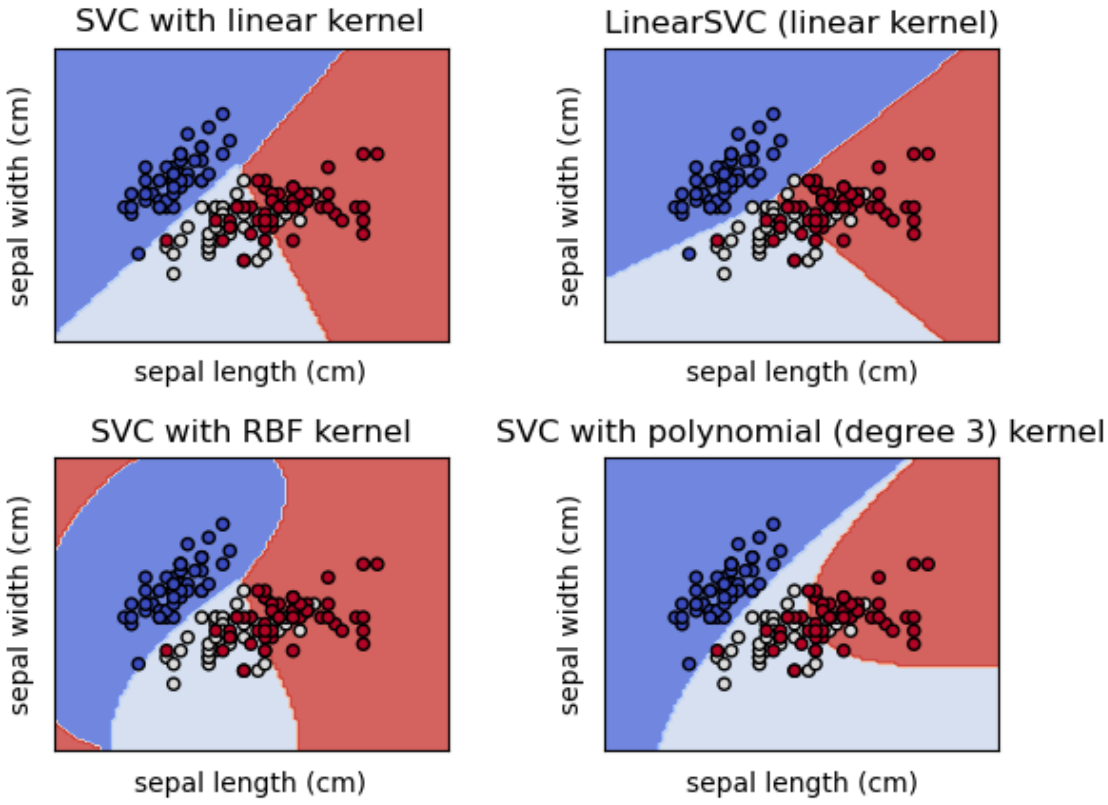

Figure 7 – Exemples de visualisation de SVM avec différentes fonctions noyaux *(scikit learn.org)*.

Sur l'image on peut voir la représentation des hyperplans du SVM avec les fonctions noyaux suivantes : linéaire, RBF et polynomial (Figure 7). Cet exemple est fait à partir

de l'expérimentation sur une base de données iris depuis le site *(scikit learn.org)*.

Ainsi, dans le cas de combinaison avec InceptionV3 freeze, ce n'est pas un changement de la sortie de ce CNN qui est appliqué, mais la vectorisation des sorties de modules Inception. Nous pouvons alors nous demander comment le SVM est entrainé. Pour effectuer ce dernier, l'architecture InceptionV3 est en état de *"Freezing"*. Cela correspond à la fixation des paramètres qui sont en amont du SVM. Ainsi, le SVM est entrainé sans créer de calculs trop longs lors de l'entrainement de InceptionV3.

De plus, une comparaison sera faite avec l'entraînement de InceptionV3 en modifiant la sortie qui sera alors adaptée pour trois classes uniquement.

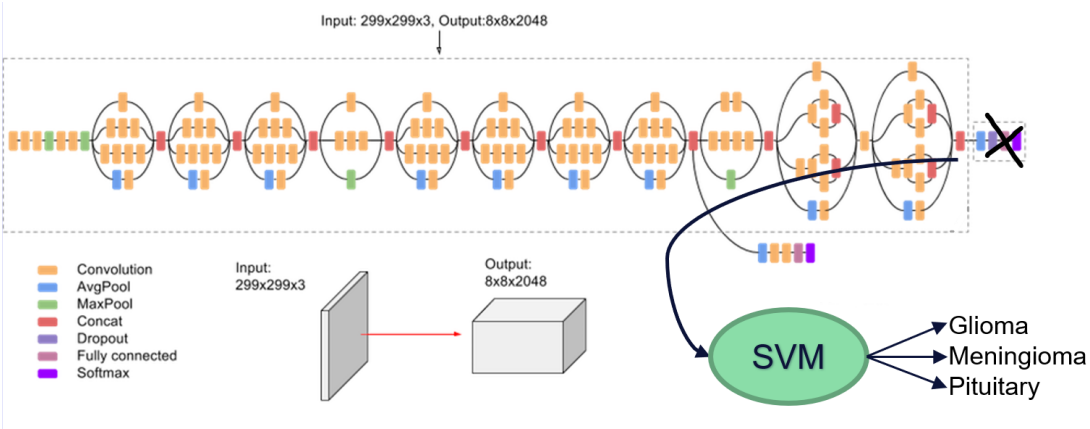

Figure 8 – Combinaison de InceptionV3 avec un SVM pour la classification inspiré de Rehman et al. (2020).

Sur la figure ci-dessus (Figure 8), nous pouvons voir quels modules Inception sont utilisés pour le SVM. C'est à partir de chaque sortie de ces modules que le SVM va faire une classification. Cela permet ainsi, de savoir à quel "étage" de l'architecture, les caractéristiques extraites sont les plus fiables pour le SVM.

3 Méthodologie expérimentale

Cette section a pour but de développer la méthodologie expérimentale de ce projet en présentant le protocole d'évaluation des performances, un description de la base de données utilisée ainsi que les différentes mesures effectuées sur les modèles.

3.1 Protocole d'évaluation des performances

Dans le but d'évaluer les performances de mes modèles, je vais utiliser les mesures de temps d'entrainement et d'inférence, les ressources utilisées ainsi que différentes métriques présenté dans la section *Mesures de performances*.

Je vais ainsi mesurer les différentes performances après avoir ajusté les hyperparamètres pour chacun des modèles, tel que le nombre maximal d'epoch, l'early-stopping, le momentum, l'utilisation ou non d'augmentation de données et la répartition de ces données.

La totalité des tests seront réalisés sous environnement Google Colab. Cela permet ainsi d'avoir la même configuration afin d'effectuer les tests, tout en y intégrant une limitation matérielle.

3.2 Description de la base de données

La base de données utilisée sera la même pour tous les modèles. Elle est disponible sur Figshare, grâce à Cheng et al. (2015). Elle comprend 3064 coupes IRM avec 708 coupes de méningiome, 1426 pour le gliome et 930 pour la tumeur pituitaire.

Cette base de données est par la suite augmentée, avec les transformations suivantes :

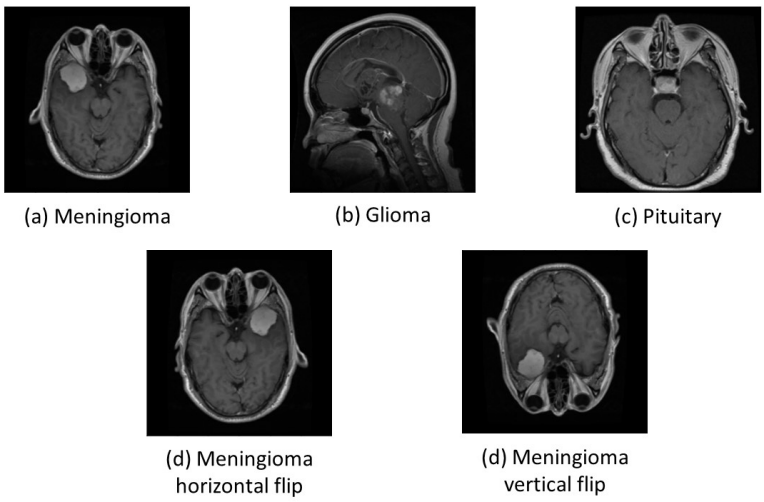

Figure 9 – Exemples d'images de la base de données ((d) et (e) font suite à l'augmentation de données.)

horizontal flip et vertical flip. Le but de cette augmentation est de palier à la faible présence d'images de meningiome et de tumeur pituitaire face aux nombres d'images de gliome.

Au cours du projet, il y aura deux répartitions différentes pour réaliser les tests. Dans un premier temps, tous les modèles sont testés sans augmentation de données. La **répartion initiale** est donc la suivante :

— Base d'entraînement : 72% (2207 images)

— Base de validation : 18% (550 images)

— Base de test : 10% (307 images)

Puis la **répartition** est modifiée suite à l'**augmentation de données** afin d'obtenir une plus grande proportion de données d'entraînement. La répartion devient ainsi :

— Base d'entraînement : 76.4% (7031 images)

— Base de validation : 13.5% (1241 images)

— Base de test : 10% (920 images)

3.3 Mesures de performances

Temps d'entraînement

Google Colab permet de coder sous forme de bloc, à l'instar de Jupyter Notebook. Ce fonctionnement par bloc, permet au sein de l'application de connaître la durée d'exécution de chacun des blocs. C'est à partir des durées fournies par Google Colab que je me baserai pour connaître le temps d'entraînement des modèles.

Temps d'inférence

Sur le même principe que le temps d'entraînement, je vais utiliser les durées cette fois-ci des blocs de prédictions pour connaître le temps d'inférence de chacun des modèles.

Ressources utilisées

Google Colab permet de connaître les ressources utilisées suite à l'éxécution d'un programme. Les données disponibles sont la mémoire vive utilisée (RAM), la mémoire du GPU utilisée et l'espace occupé sur le disque.

Métriques de performances

Une matrice de confusion est un outil utilisé en apprentissage automatique pour évaluer la performance d'un modèle de classification. Elle permet de visualiser les prédictions faites par le modèle et de les comparer aux valeurs réelles (appelées labels) pour chaque exemple de données.

Une matrice de confusion est une matrice carrée de n lignes et n colonnes, où n est le nombre de classes possibles dans les données. Chaque ligne de la matrice représente une classe réelle, tandis que chaque colonne représente une classe prédite par le modèle.

Voici un exemple de matrice de confusion pour un modèle de classification binaire :

	Prédit comme positif	Prédit comme négatif
Réellement positif	True Positive (TP)	False Negative (FN)
Réellement négatif	False Positive (FP)	True Negative (TN)

Table 1 – Exemple de matrice de confusion pour un modèle binaire

On peut ensuite utiliser cette matrice pour calculer différentes mesures de performance du modèle, telles que l'accuracy (8), la précision (9) et le taux de vrais positifs (recall)(10).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

L'accuracy correspond au ratio entre le nombre de bonnes prédictions sur le nombre de prédictions totales. Ce score s'exprime entre 0 et 1. Egalement, si les faux positifs (FP) et les faux négatifs (FN) se font rare, alors ce score tend vers 1.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

La sensibilité (ou précision) est une mesure de performance du modèle qui indique combien de fois il a correctement prédit la classe positive parmi tous les éléments qu'il a prédit comme appartenant à cette classe. Ce score s'exprime entre 0 et 1. Egalement, si les faux positifs (FP) se font rare, alors ce score tend vers 1.

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

Le recall (ou spécificité) est une mesure de performance du modèle qui indique combien de fois il a correctement prédit les éléments appartenant à une classe particulière. Ce score

s'exprime entre 0 et 1. Egalement, si les faux négatifs (FN) se font rare, alors ce score tend vers 1.

Il est important de noter que la matrice de confusion n'est utile que pour les modèles de classification, et qu'elle ne peut pas être utilisée pour évaluer les performances de modèles de régression ou de prédiction de séquences.

Les matrices de confusions permettent également de tracer des courbes ROC. Ces courbes permettent de déterminer la qualité d'un classificateur.

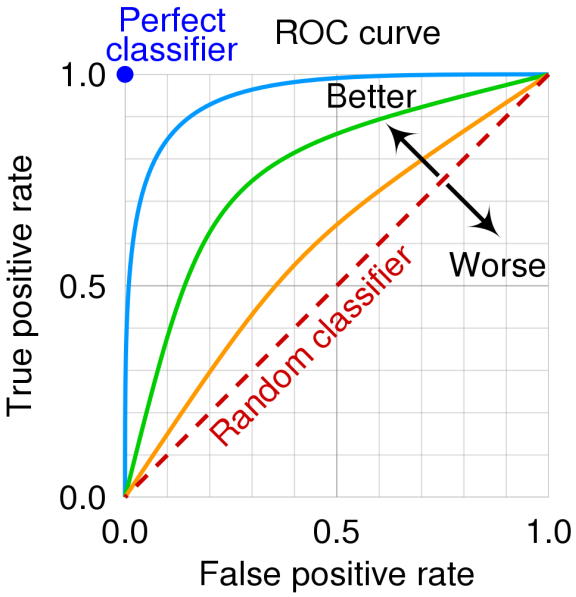

Figure 10 – Illustration de la courbe ROC pour un classificateur (MartinThoma, 2018)

4 Résultats de simulation

4.1 Représentation des résultats

CNN sans nom

Nombre total de paramètre : 16,809,059 — Nombre de paramètre à entraîner 16,808,995

Ressources nécessaires pour entraîner le modèle :

— RAM : 6.02GB (7.72GB avec augmentation de données)

— GPU : 9.60GB (10.02GB avec augmentation de données)

Augmentation de données	Non	Oui
Learning rate	0.0001	0.0001
Momentum	0.9	0.5
Temps d'entraînement	1h 16min	2h 23min
Temps d'inférence	15ms	13ms
Training loss	0.0692	0.0516
Validation loss	0.3164	0.2711
Test loss	0.3513	0.1772
Accuracy	89.87%	93.15%
Nombre d'epoch avant early stopping	20	12

Table 2 – Résultats pour CNN sans nom

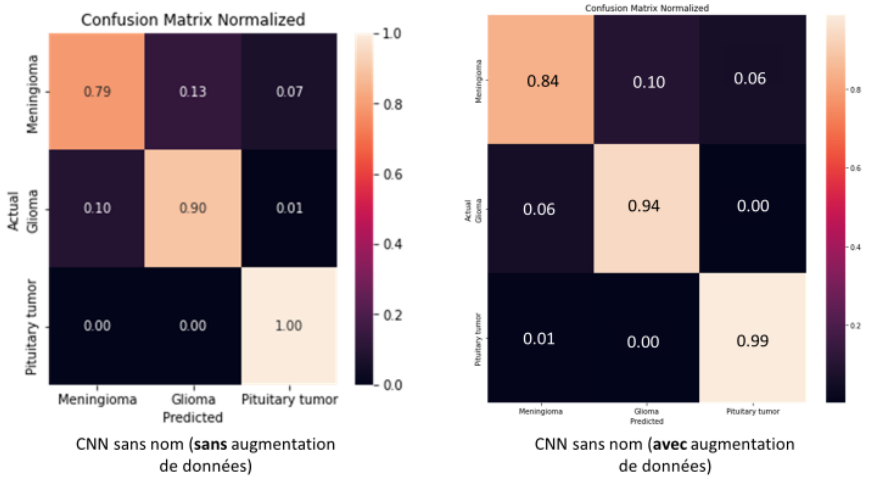

Figure 11 – Matrices de confusion pour le CNN sans nom.

InceptionV3 sans pré-entraînement

Nombre total de paramètre : 21,808,961 — Nombre de paramètre à entraîner : 21,774,529

Ressources nécessaires pour entraîner le modèle :

— RAM : 5.6GB (7.60GB avec augmentation de données)

— GPU : 7.62GB (7.89GB avec augmentation de données)

Augmentation de données	Non	Oui
Learning rate	0.0001	0.0001
Momentum	0.8	0.5
Temps d'entraînement	50min 10s	56min 12s
Temps d'inférence	10ms	8ms
Training loss	0.2034	0.2060
Validation loss	0.2120	0.1662
Test loss	0.2012	0.1415
Accuracy	94.81%	95.11%
Nombre d'epoch avant early stopping	48	16

Table 3 – Résultats pour InceptionV3 sans pré-entraînement

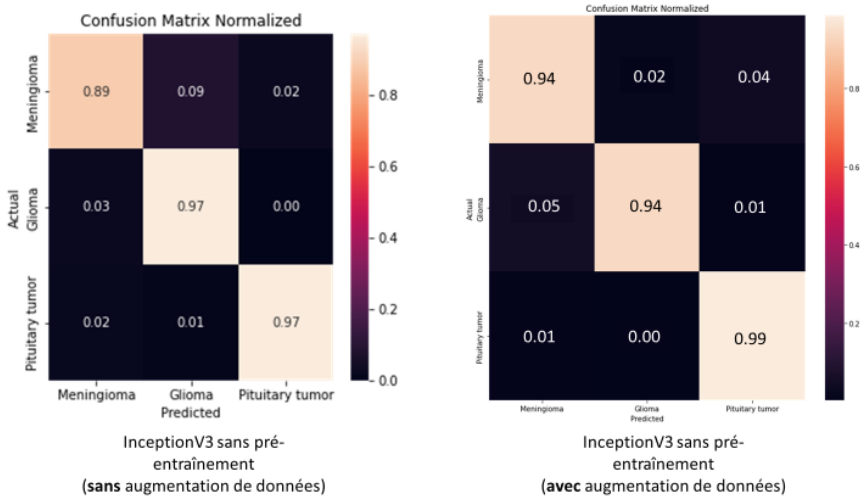

Figure 12 – Matrices de confusion pour InceptionV3 sans pré-entraînement.

InceptionV3 freeze et SVM en sortie

Nombre total de paramètre : 22,196,033 — Nombre de paramètre à entraîner : 393,249

Stockage nécessaire pour les poids du modèle pré-entrainé : 83.3 Mo.

Ressources nécessaires pour entraîner le modèle :

— RAM : 5.4GB (7.58GB avec augmentation de données)

— GPU : 8.59GB (8.99GB avec augmentation de données)

Augmentation de données	Non	Oui
Temps d'entraînement	57min 13s	2h 53min
Temps d'inférence	7ms	6ms
Training loss	0.2541	0.1149
Validation loss	2.4356	1.2639
Test loss	2.5349	1.0407
Accuracy	86.64%	88.91%
Nombre d'epoch avant early stopping	85	100

Table 4 – Résultats pour InceptionV3 freeze + SVM

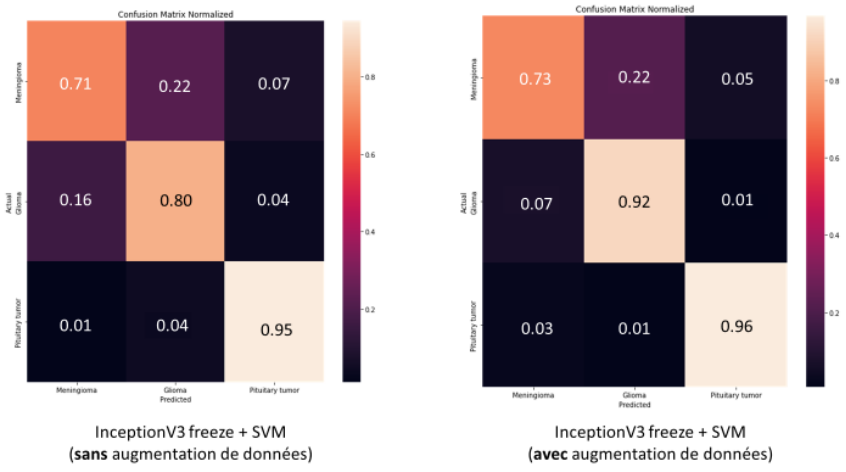

Figure 13 – Matrices de confusion pour InceptionV3 freeze avec SVM.

4.2 Interprétation des résultats

Les modèles utilisent à peu près les mêmes ressources en RAM et GPU pour s'entraîner. J'ai bien évidement dû adapter le modèle du CNN sans nom tiré de Sadoon and Ali (2021). Sans cette adaption, le modèle ne pouvait pas être entraîner à l'aide de Google Colab.
En comparant les évolutions de chacun des modèles suite à l'augmentation de données, on comprends rapidement son rôle : limiter le sur-apprentissage et également améliorer la généralisation. Cette augmentation a permis l'amélioration de l'accuracy de chacun des modèles et également de réduire les pertes lors du calcul de la fonction de coût. Je constate également que le modèle InceptionV3 sans pré-entraînement est celui qui augmente le moins son accuracy lors de la phase de test. Cela montre, que suite à l'augmentation de donnée, ce modèle se rapproche très grandement de la limite qu'il peut atteindre.

Suite à ma synthèse de littérature, je m'attendais à observer des confusions pour la prédiction du méningiome. Cette confusion semble en effet bien présente sur les trois modèles présentéees, que cela soit avec la base de données initiales ou avec la base augmentée. Le recall pour le meningiome est le plus bas parmis tous les modèles.

Actuellement, le modèle qui présente les meilleurs résultats est le modèle InceptionV3 sans pré-entraînement. Il présente le meilleur taux d'accuracy dans le cas initial et également suite à l'augmentation des données.
En ce qui concerne le nombre d'epoch, c'est le modèle du CNN sans nom qui en nécessite le moins avant d'être arrêter par l'early stopping. Il serait très interessant, de relancer les

modèles sur un nombre d'epoch plus élevé afin de vérifier si l'early stopping jouait bien son rôle, c'est à dire arrêter l'entraînement avant le sur-apprentissage. Le risque avec l'utilisation de Google Colab est la durée maximale d'utilisation qui varie aléatoirement. Lorsque l'entraînement dépasse 2-3h, il y de grandes chances que la session se fasse déconnecter. C'est la raison pour laquelle j'ai paramétré les entraînements pour qu'ils dépassent pas cette limite de temps.

Les temps d'inférence des modèles sont tous du même ordre de grandeur, environ 10ms par image. Les modèles semblent très rapides dans le rôle de classificateur.

En ce qui concerne la rapidité d'entraînement, le modèle InceptionV3 freeze avec un SVM est le plus long, l'utilisation du freeze permet de limiter le nombre de paramètres à apprendre, cependant, le SVM est très long à entraîner. L'amélioration entre chaque époch devient très minime lorsque l'on dépasse 10 époch.

Le modèle CNN sans nom possèdent moins de paramètres que les autres modèles, il est donc plus léger, mais sa structure ne permet pas un apprentissage aussi optimisé que pour InceptionV3 sans pré-entraînement.

Ainsi, le modèle InceptionV3 sans pré-entraînement est le modèle le performant pour classer trois types de tumeurs à partir d'IRM du cerveau. Il est également le modèle le plus rapide à entraîner.

5 Conclusion

5.1 Recommandations pour la suite du projet

En ce qui concerne les recommandations pour la suite de ce projet, il faudrait utiliser du matériel plus puissant que ce que propose Google Colab afin de pouvoir faire une augmentation de données plus importantes en implémentant du bruit dans certaines images afin d'augmenter la robustesse des modèles.

De plus, j'ai pu implémenté les matrices de confusions, mais je n'ai pas eu le temps de mettre en place les courbes ROC. Ces dernières permettraient de finaliser mon projet et de comparer la sensibilité face à la spécificité des modèles.

Lors de la réalisation de ce projet, j'aurai également voulu tester la robustesse des modèles. L'implémentant d'une seconde base de données aurait permis de pouvoir tester les modèles sur leurs processus de généralisation et ainsi leurs robustesses.

5.2 Conclusions principales

Le cours sur les réseaux de neurones m'a aidé à mieux comprendre comment ils fonctionnent. Il s'intègre bien dans ma formation d'ingénieur en robotique et en santé. Les réseaux de neurones de plus en plus utilisé dans de nombreux domaines, tels que la reconnaissance de formes et d'objets, la santé et la robotique, l'éducation.

J'aimerai par la suite pouvoir lier le domaine de la santé à la robotique, et l'utilisation de réseaux de neurones est une des plus grandes passerelles actuelles. Le sujet étudié à

travers ce projet, constitue un exemple de ce qui est encore recherché aujourd'hui dans le domaine médical.

Les résultats expérimentaux obtenus n'atteignent pas les performances promises dans les recherches utilisées pour la synthèse de littérature. Cependant, ces derniers sont très proche de ce qui était attendu. Il est donc judicieux de poursuivre l'entraînement de ces derniers afin d'atteindre un ratio coût/performance qui dépasserait mes lectures.

Références

J. Cheng, W. Huang, S. Cao, R. Yang, W. Yang, Z. Yun, Z. Wang, and Q. Feng. Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition. *PLOS ONE*, 10(10) :e0140381, Oct. 2015. ISSN 1932-6203. doi : 10.1371/journal.pone.0140381. URL https://dx.plos.org/10.1371/journal.pone.0140381.

P. Chlap, H. Min, N. Vandenberg, J. Dowling, L. Holloway, and A. Haworth. A review of medical image data augmentation techniques for deep learning applications. *Journal of Medical Imaging and Radiation Oncology*, 65(5) :545–563, 2021. ISSN 1754-9485. doi : 10.1111/1754-9485.13261. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1754-9485.13261. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1111/1754-9485.13261.

Google. Guide avancé d'Inception v3 | Cloud TPU, na. URL https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=fr.

c. MartinThoma. English : Receiver Operating Characteristic (ROC) curve with False Positive Rate and True Positive Rate. The diagonal shows the performance of a random classifier. Three example classifiers (blue, orange, green) are shown. Drawn by CMG Lee based on http ://commons.wikimedia.org/wiki/File :roc-draft-xkcd-style.svg ., June 2018. URL https://commons.wikimedia.org/wiki/File:Roc_curve.svg#/media/File:Roc_curve.svg.

oreilly.com. 4. Fully Connected Deep Networks - TensorFlow for Deep Learning [Book]. URL https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html. ISBN : 9781491980453.

A. Pashaei, H. Sajedi, and N. Jazayeri. Brain Tumor Classification via Convolutional Neural Network and Extreme Learning Machines. In *2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 314–319, Oct. 2018. doi : 10.1109/ICCKE.2018.8566571. ISSN : 2375-1304.

A. Rehman, S. Naz, M. I. Razzak, F. Akram, and M. Imran. A Deep Learning-Based Framework for Automatic Brain Tumors Classification Using Transfer Learning. *Circuits, Systems, and Signal Processing*, 39(2) :757–775, Feb. 2020. ISSN 1531-5878. doi : 10.1007/s00034-019-01246-3. URL https://doi.org/10.1007/s00034-019-01246-3.

T. A. Sadoon and M. H. Ali. Deep learning model for glioma, meningioma and pituitary classification. *International Journal of Advances in Applied Sciences*, 10(1) :88, Mar. 2021. ISSN 2722-2594, 2252-8814. doi : 10.11591/ijaas.v10.i1.pp88-98. URL http://ijaas.iaescore.com/index.php/IJAAS/article/view/20439.

scikit learn.org. Plot different SVM classifiers in the iris dataset. URL https://scikit-learn/stable/auto_examples/svm/plot_iris_svc.html.

A. Srinivasa Reddy and P. Chenna Reddy. MRI brain tumor segmentation and prediction using modified region growing and adaptive SVM. *Soft Computing*, 25(5) :4135–4148, Mar. 2021. ISSN 1432-7643, 1433-7479. doi : 10.1007/s00500-020-05493-4. URL http://link.springer.com/10.1007/s00500-020-05493-4.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision, Dec. 2015. URL http://arxiv.org/abs/1512.00567.