

Rapport Projet TIP

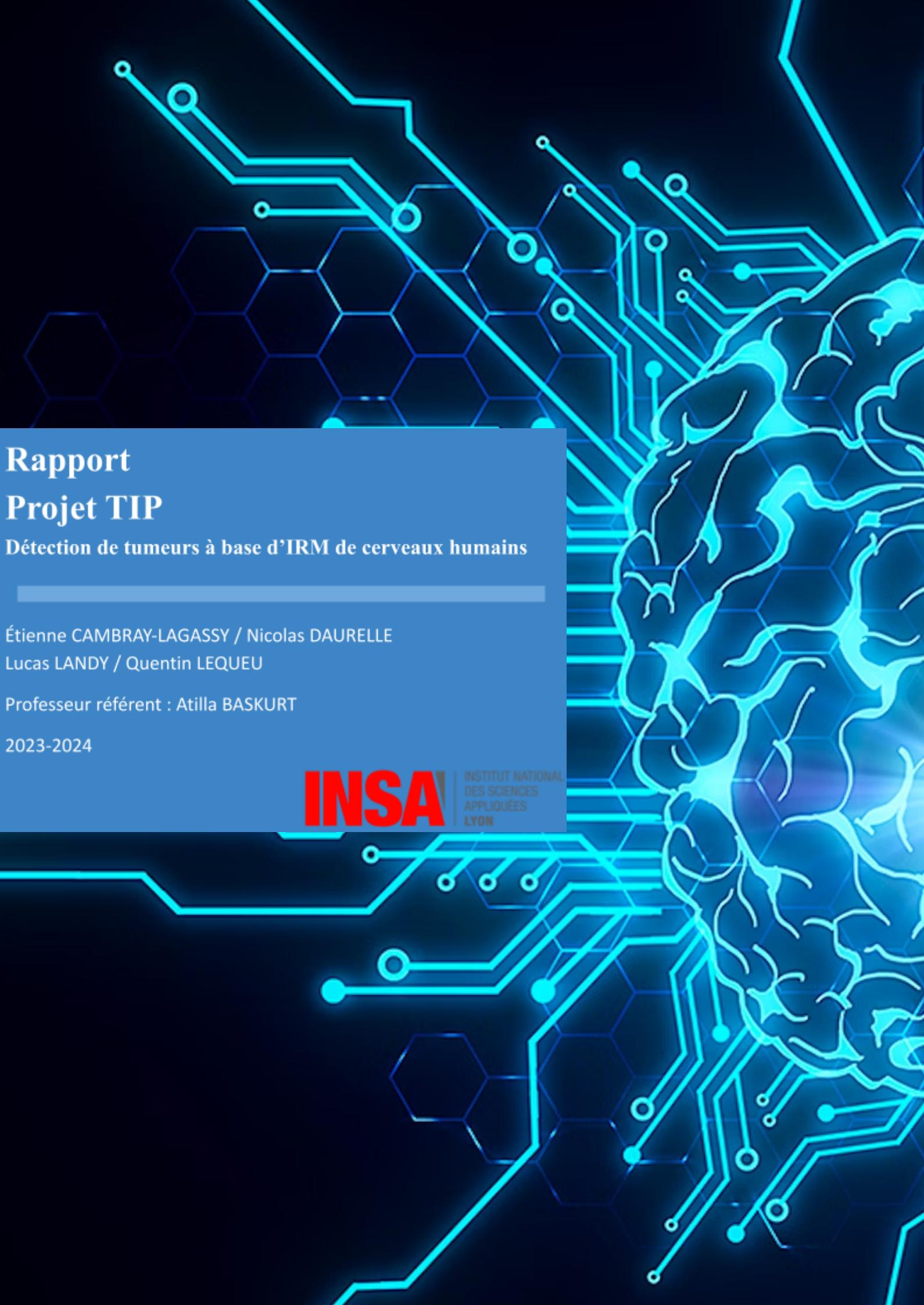
Détection de tumeurs à base d'IRM de cerveaux humains

Étienne CAMBRAY-LAGASSY / Nicolas DAURELLE

Lucas LANDY / Quentin LEQUEU

Professeur référent : Atilla BASKURT

2023-2024



Rapport du projet Traitement d'Images et Paroles

SOMMAIRE

Introduction

Partie I : Prémices du projet

1. Recherches
2. Données

Partie II : Mise en place du projet

1. Mise en place d'un simple réseau de neurone convolutif
2. Résultat obtenu lors de la première version du projet

Partie III : Axes d'améliorations du projet

1. Améliorations des performances
 - a. Augmentation du jeu de données
 - b. Amélioration du réseau de neurone afin d'éviter le surapprentissage
 - c. Filtrage
 - d. Variations du réseau de neurones
2. Perspectives d'améliorations
 - a. Délimiter la zone de la tumeur
 - b. Utiliser un filtrage performant

Conclusion

Annexe

- Sources
- Informations
- Outils utilisés
- Tâches effectuées
- Lexiques

Introduction

Notre projet consiste à détecter la présence de tumeur dans des IRM de cerveaux humains à l'aide d'un réseau de neurones convolutif. Nous avons choisi ce sujet car nous étions tous intéressés par un projet traitant de l'imagerie médicale. Ce projet a été réalisé en environ 30h d'études, de développements et de tests, mais aussi 5h de rédaction et de création du support de présentation.

Le but était, au départ, de détecter si un cerveau était atteint d'une tumeur ou non (il existait donc à l'origine 2 classes), mais nous avons ensuite fait évoluer notre projet afin de diviser la classe des tumeurs en 3 classes qui correspondent à 3 types de tumeurs : Meningioma, Glioma, Pituitary (ainsi que la 4ème classe, l'absence de tumeur). Nous présenterons ici les différentes étapes du projet et de notre programme.

Ce projet s'est développé au cours de plusieurs semaines et se divise en 4 phases de développement :

- La première phase est une phase de recherche documentaire, pour nous orienter vers une méthode de réalisation, ainsi que de recherche de données exploitables.
- La deuxième phase est le développement d'une version primaire du programme ne comportant que 2 classes.
- La troisième phase est l'évolution de notre code afin que le réseau de neurones range les images d'IRM parmi 4 classes différentes.
- La dernière phase est une phase d'optimisation de notre projet, qui nous a permis de gagner en précision.

Partie I : Prémices du projet

1. Recherches

Avant de débuter notre projet, il était primordial pour nous d'effectuer des recherches pour déterminer les possibles axes que nous pouvions explorer afin de réaliser ce projet.

Tout d'abord, nous avons recherché des documents scientifiques qui nous permettraient d'en savoir davantage sur des solutions utilisées actuellement pour détecter des tumeurs dans un cerveau. Nous nous sommes attardés sur le document intitulé "*Détection du cancer du cerveau à l'aide d'un réseau de neurones*"^{*3} publié en 2019 par M. EL KOUCH et M. OTMAN. Pour la réalisation du projet, nous nous sommes appuyés sur certaines informations de ce papier comme le fait que nous pouvions utiliser des images en niveaux de gris, mais aussi l'utilisation de la méthode GLCM pour l'analyse de texture en guise de voie d'amélioration.

Suite à cela, nous nous sommes intéressés à des vidéos disponibles sur Youtube qui pourraient nous aider à réaliser ce projet. Nous nous sommes attardés sur la vidéo intitulée "*Coder un réseau de neurones convolutifs de classification d'image avec Python et Tensorflow*"^{*2} réalisé par Defend Intelligence. Cette vidéo nous a permis d'en apprendre davantage sur l'utilisation de Python et Tensorflow et d'obtenir une base de réseau de neurones convolutif.

D'autres recherches ont été effectuées, cela concerne principalement l'utilisation de la librairie Tensorflow pour l'utilisation, la réalisation et l'amélioration d'un réseau de neurones convolutifs sur Python.

2. Données

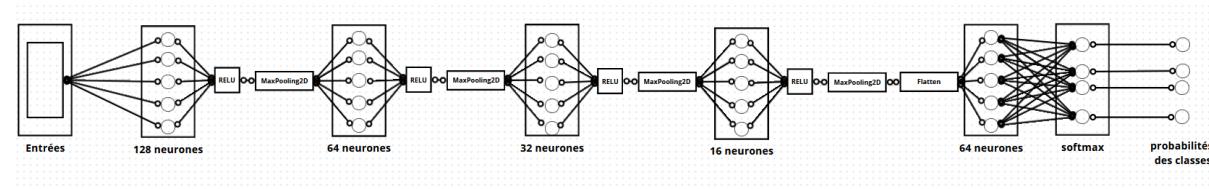
Afin de réaliser notre projet, nous nous sommes attardés sur le choix de dataset^{*4} accessibles sur Kaggle. Pour choisir nos datasets, nous nous sommes concentrés sur la qualité de ces derniers. En effet, il existe une multitude de dataset d'IRM de cerveaux qui regroupent les tumeurs que nous désirions, mais certaines bases de données possèdent un indice d'utilisation trop faible, un frein pour le bon fonctionnement de notre projet.

Pour ce début de projet, nous nous sommes donc appuyés sur 2 datasets afin de regrouper 1176 images pour l'entraînement de notre réseau de neurones et 529 images pour la vérification. Ces datasets^{*4} possèdent un indice d'intérêt équivalent à la note de 8,75/10 attribuée par Kaggle.

Partie II : Mise en place du projet

1. Mise en place d'un simple réseau de neurone convolutif

Afin d'expliquer convenablement le fonctionnement de notre réseau de neurones, il est important de faire son schéma.



Capture n°1 : Schéma du premier réseau de neurone

Ce réseau a été réalisé grâce à un code python en utilisant la librairie TensorFlow, utilisé pour la création de réseau de neurones. Il s'appuie sur la vidéo Youtube citée plus tôt.

```
#NOMBRE DE CLASSE 4, CELA CORRESPOND A NOS 3 TUMEURS DIFFERENTES ET L'ABSENCE DE TUMEUR
num_classes = 4

model = tf.keras.Sequential([
    #NORMALISER LES COULEURS POUR FAIRE QUE LES PIXELS SOIT COMPRIS ENTRE 0 ET 1
    #PERMET LA CONVERGENCE PLUS RAPIDE LORS DE L'APPRENTISSAGE DU MODELE
    layers.experimental.preprocessing.Rescaling(1./255),
    #COUCHE CONVOLUTIVE DE 128 NEURONES AVEC RELU
    #LA VALEUR 4 CORRESPOND AU FILTRE QUI SE BALADE SUR L'IMAGE DE FORMAT 4*4
    #RELU PERMET D'EVITER LA NON-LINEARITE EN PASSANT LES IMAGES NEGATIVES A 0
    layers.Conv2D(128, 4, activation='relu'),
    layers.MaxPooling2D(),
    #COUCHE CONVOLUTIVE DE 64 NEURONES AVEC RELU
    layers.Conv2D(64, 4, activation='relu'),
    layers.MaxPooling2D(),
    #COUCHE CONVOLUTIVE DE 32 NEURONES AVEC RELU
    layers.Conv2D(32, 4, activation='relu'),
    layers.MaxPooling2D(),
    #COUCHE CONVOLUTIVE DE 16 NEURONES AVEC RELU
    layers.Conv2D(16, 4, activation='relu'),
    #CREATION D'UNE COUCHE DE POOLING :
    #REDUIT LES DIMENSIONS SPATIALES DE LA REPRESENTATION DE L'IMAGE EN SELECTIONNANT VMAX D'UNE REGION DONNEE
    layers.MaxPooling2D(),
    #CONVERTIR LES DONNEES D'ENTREE BIDIMENSIONNELLES EN UN VECTEUR UNIDIMENSIONNEL
    layers.Flatten(),
    #COUCHE CONVOLUTIVE DE 64 NEURONES AVEC RELU
    layers.Dense(64, activation='relu'),
    #SOFTMAX FONCTION D'ACTIVATION DE DERNIERE COUCHE POUR MULTI-CLASSES
    layers.Dense(num_classes, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Capture n°2 : Code python qui permet la création du réseau de neurones

L'ensemble des explications de ce code est accessible auprès de la capture ci-dessus grâce aux commentaires.

```
*** Epoch 1/100
1255/1255 [=====] - 190s 150ms/step - loss: 1.1605 - accuracy: 0.4416 - val_loss: 1.0857 - val_accuracy: 0.4511
Epoch 2/100
1255/1255 [=====] - 188s 150ms/step - loss: 1.0664 - accuracy: 0.4811 - val_loss: 1.0119 - val_accuracy: 0.4766
Epoch 3/100
1255/1255 [=====] - 187s 149ms/step - loss: 0.9650 - accuracy: 0.5452 - val_loss: 0.8559 - val_accuracy: 0.6202
Epoch 4/100
1255/1255 [=====] - 187s 149ms/step - loss: 0.7951 - accuracy: 0.6514 - val_loss: 0.7423 - val_accuracy: 0.6926
Epoch 5/100
211/1255 [====>.....] - ETA: 2:30 - loss: 0.7372 - accuracy: 0.6746
```

Capture n°3 : Première phase d'entraînement du réseau

Lors de l'entraînement du réseau, nous obtenons le résultat ci-dessus. Il regroupe les informations primordiales comme le nombre d'epochs réalisées, le pourcentage de détermination moyen avec la valeur “accuracy”, qui correspond au taux de réussite pour déterminer à quelles classes appartiennent les images avec lesquelles il s'entraîne. On y voit aussi la “val_accuracy” qui correspond au pourcentage de réussite de déterminer la classe d'une image non utilisée lors de l'entraînement. Dans notre cas, à la seconde epoch, la détermination est de 50% pour les images d'entraînements et les images de validations.

2. Résultat obtenu lors de la première version du projet

Afin de déterminer si notre réseau de neurones est efficace ou non, il était important pour nous de le mesurer et obtenir un aperçu de son efficacité. Nous avons ainsi obtenu les résultats ci-dessous. Grâce à cela, nous obtenons le pourcentage de réussite moyen pour une bonne détection et nous obtenons le pourcentage de réussite en fonction de chacune de nos quatres classes.

Le pourcentage d'erreur pour une tumeur glioma est : 20.0%
Le pourcentage de réussite pour une tumeur glioma est : 80.0%

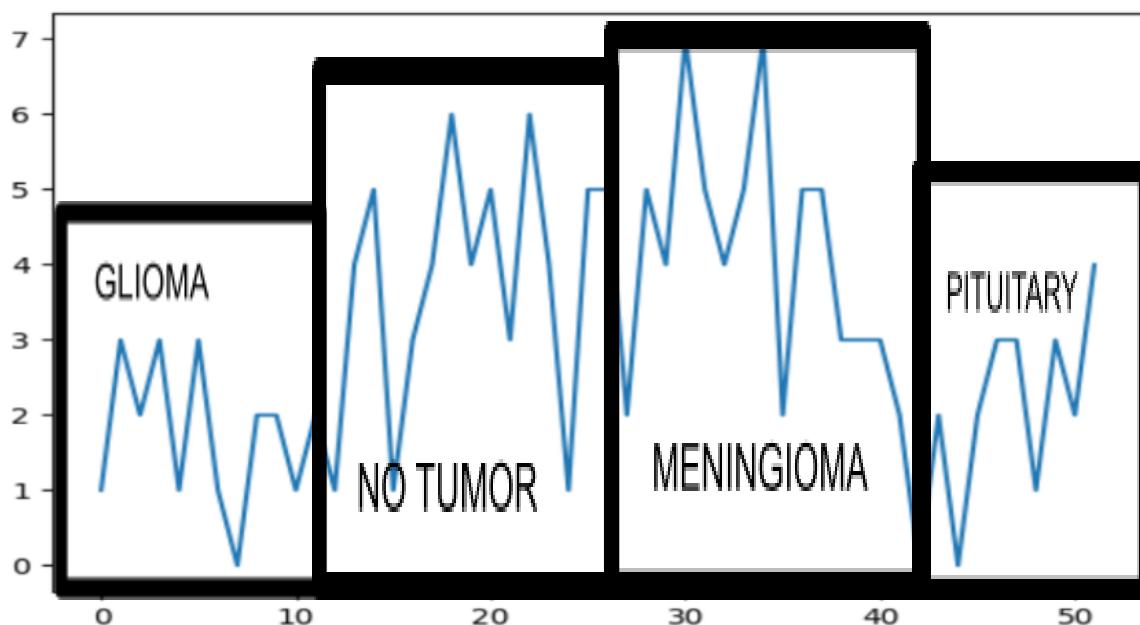
Le pourcentage d'erreur pour une l'absence de tumeur est : 44.61538461538462%
Le pourcentage de réussite pour une l'absence de tumeur est : 55.38461538461538%

Le pourcentage d'erreur pour une tumeur Menigioma est : 36.11111111111111%
Le pourcentage de réussite pour une tumeur Menigioma est : 63.8888888888889%

Le pourcentage d'erreur pour une tumeur Pituitary est : 21.379310344827587%
Le pourcentage de réussite pour une tumeur Pituitary est : 78.62068965517241%

The medium error rate is : 30.52645151783083%
The medium success rate is : 69.47354848216918%

Capture n°4 : Entraînement de notre réseau de neurones



Capture n°5 - Mesures des résultats obtenus après l'utilisation de notre jeu de test auprès de notre réseau de neurones convolutifs

Le schéma correspond à la détection du nombre d'erreurs tous les 10 échantillonnages. Il nous permet d'obtenir un réel aperçu pour ce qui est de la différence d'efficacité, lorsque nous passons d'une classe à une autre.

Ces résultats démontrent l'efficacité de notre réseau de neurones en fonction des classes. Ainsi, nous obtenons un pourcentage de réussite de 80% pour la tumeur Glioma, 63% pour la tumeur Meningioma et 78% pour la tumeur Pituitary. Nous pourrions nous dire que les résultats sont convenables pour l'absence de complexité au sein de ce réseau de neurones, sauf qu'il existe une dernière classe, l'absence de tumeur, où les résultats de réussites sont assez mauvais à savoir 55% de réussite. Cela peut se traduire par le fait qu'une personne sans tumeur a 50% de chance de se voir diagnostiquer une tumeur alors qu'elle n'en a pas.

Partie III : Axes d'améliorations du projet

1. Améliorations des performances

a. Augmentation du jeu de données

Le premier axe d'amélioration que nous avons réalisé afin d'accroître la précision de notre réseau était d'augmenter le volume de données traitées. Nous avons trouvé 2 jeux de données^{*5} en plus comportant chacun entre 500 et 1000 images de chaque classe étudiée, les tumeurs et l'absence de cancer. Chaque jeu comportait un indice d'intérêt correct, 7.50 pour le premier et 8.13 pour le deuxième. Pour chaque jeu, environ deux tiers des images ont été ajoutées aux données d'entraînement et un tiers a été ajouté aux données de tests.

Le pourcentage d'erreur pour une tumeur glioma est : 56.56063618290258%

Le pourcentage de réussite pour une tumeur glioma est : 43.43936381709742%

Le pourcentage d'erreur pour une l'absence de tumeur est : 60.47120418848168%

Le pourcentage de réussite pour une l'absence de tumeur est : 39.52879581151832%

Le pourcentage d'erreur pour une tumeur Menigioma est : 36.58838071693449%

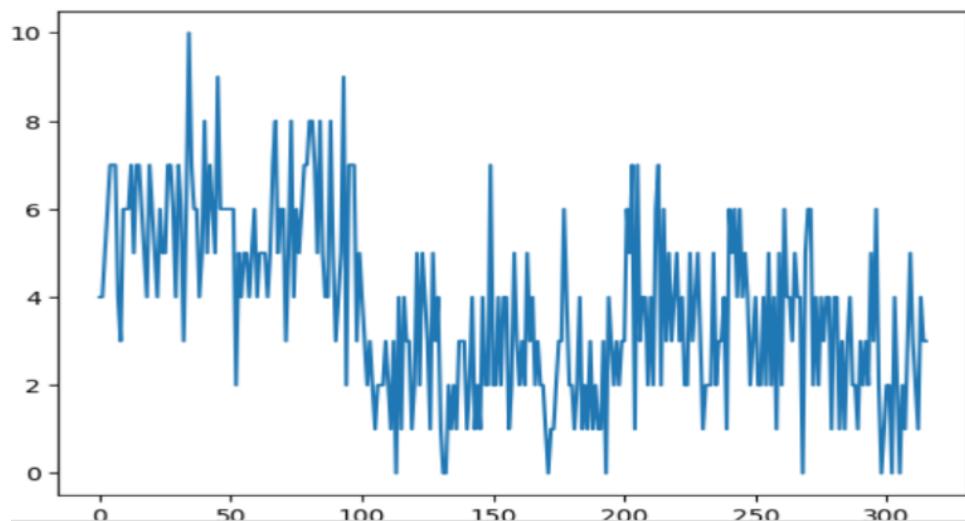
Le pourcentage de réussite pour une tumeur Menigioma est : 63.41161928306551%

Le pourcentage d'erreur pour une tumeur Pituitary est : 9.854771784232366%

Le pourcentage de réussite pour une tumeur Pituitary est : 90.14522821576763%

The medium error rate is : 40.868748218137775%

The medium success rate is : 59.131251781862225%



Capture n°6 - Mesures des résultats obtenus après l'augmentation de la taille de notre jeu d'entraînement et de test auprès de notre réseau de neurones convolutifs

Au premier abord, nous pourrions nous dire que l'ajout du nouveau jeu de données pour l'entraînement et pour les tests a seulement été néfaste pour notre détection, si nous regardons seulement le pourcentage moyen de réussite qui a drastiquement baissé de 70% à 60%. Sauf que nous pouvons constater plus en détails que le pourcentage de réussite de détection d'une tumeur Pituitary a drastiquement augmenté en dépit des autres classes. En fonction de notre jeu de données, le pourcentage d'efficacité des classes baisse ou augmente en fonction de la quantité de données étudiées. Dans cet état, notre réseau de neurones est performant pour détecter les tumeurs Pituitary.

b. Amélioration du réseau de neurone afin d'éviter le surapprentissage

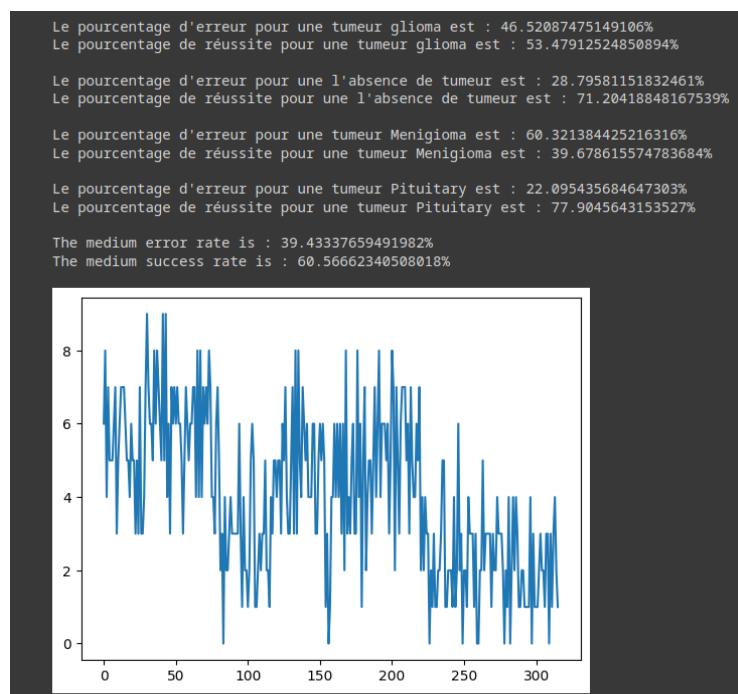
Le surapprentissage est un élément clé néfaste au bon développement de notre projet, éviter celui-ci est primordial. Cela a permis d'éviter que notre réseau s'efforce à reconnaître davantage notre jeu d'entraînement, ce qui pourrait le vouer à être moins performant lors des tests pour déterminer les classes de nos images.

```
early_stopping = EarlyStopping(monitor='val_accuracy', patience=5, restore_best_weights=True)

model = tf.keras.Sequential([
    #NORMALISER LES COULEURS POUR FAIRE QUE LES PIXELS SOIT COMPRIS ENTRE 0 ET 1
    #PERMET LA CONVERGENCE PLUS RAPIDE LORS DE L'APPRENTISSAGE DU MODELE
    layers.experimental.preprocessing.Rescaling(1./255),
    #COUCHE CONVOLUTIVE DE 128 NEURONES AVEC RELU
    #LA VALEUR 4 CORRESPOND AU FILTRE QUI SE BALADE SUR L'IMAGE DE FORMAT 4*4
    #RELU PERMET D'EVITER LA NON-LINEARITE EN PASSANT LES IMAGES NEGATIVES A 0
    layers.Conv2D(128, 4, activation='relu'),
    #DROPOUT PERMET D'EVITER LE SURAPPRENTESSAGE
    layers.Dropout(0.05, noise_shape=None, seed=None),
    layers.MaxPooling2D(),
    #COUCHE CONVOLUTIVE DE 64 NEURONES AVEC RELU
    layers.Conv2D(64, 4, activation='relu'),
    layers.Dropout(0.05, noise_shape=None, seed=None),
    layers.MaxPooling2D(),
    #COUCHE CONVOLUTIVE DE 32 NEURONES AVEC RELU
    layers.Conv2D(32, 4, activation='relu'),
    layers.Dropout(0.05, noise_shape=None, seed=None),
    layers.MaxPooling2D(),
    #COUCHE CONVOLUTIVE DE 16 NEURONES AVEC RELU
    layers.Conv2D(16, 4, activation='relu'),
    #CREATION D'UNE COUCHE DE POOLING :
    #REDUIT LES DIMENSIONS SPATIALES DE LA REPRESENTATION DE L'IMAGE EN SELECTIONNANT VMAX D'UNE REGION DONNEE
    layers.MaxPooling2D(),
    #CONVERTIR LES DONNEES D'ENTREE BIDIMENSIONNELLES EN UN VECTEUR UNIDIMENSIONNEL
    layers.Flatten(),
    #COUCHE CONVOLUTIVE DE 64 NEURONES AVEC RELU
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.05, noise_shape=None, seed=None),
    #SOFTMAX FONCTION D'ACTIVATION DE DERNIERE COUCHE POUR MULTI-CLASSES
    layers.Dense(num_classes, activation='softmax')
])
```

Capture n°7 - Configuration du premier réseau de neurones pour éviter le surapprentissage

Pour cette phase, nous avons ainsi mis en place un “*early_stopping*” qui nous permet de nous arrêter lorsque notre “*val_accuracy*” diminue durant 5 epochs. Nous avons aussi mis en place l’aspect de “*layers.Dropout*” qui nous permet d’éviter le surapprentissage sur chaque couche de notre réseau. Ces deux configurations vont permettre à notre réseau de neurones de s’arrêter à temps afin d’éviter le surapprentissage. L’objectif est ainsi d’obtenir de meilleurs résultats lors de nos tests de déterminations de classes auprès de notre réseau de neurones.

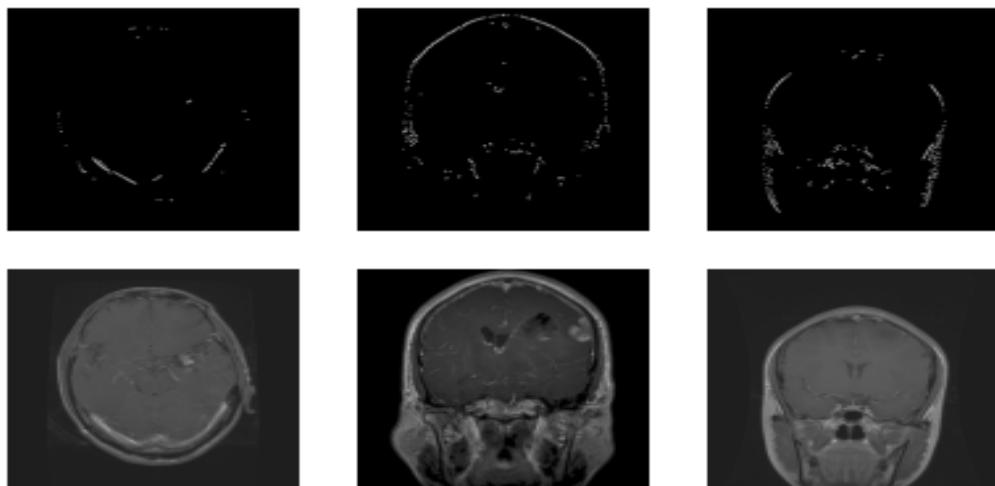


Capture n°8 - Résultat obtenu après la suppression du surapprentissage

En évitant le surapprentissage, nous avons “stagné” pour ce qui est des performances de notre réseau de neurones, mais dorénavant, celui-ci s’arrête quand il détermine de lui-même, qu’il n’est pas nécessaire de poursuivre ses entraînements. Grâce à cela, la réalisation d’epochs est moindre à savoir 8. Pour ce qui est de la détection moyenne, elle est identique. En revanche, la détection des classes a changé, nous sommes capables de détecter davantage l’absence de tumeur, ce qui n’était pas le cas avant, mais cela est en dépit de l’efficacité de détection des autres classes.

c. Filtrage

De prime abord, nous avons pensé à filtrer les images pour n'étudier que les contours, cependant cette voie s'est avérée être une impasse car les tumeurs se différencient par leurs textures, qui ne sont pas visibles si nous analysons seulement les contours.

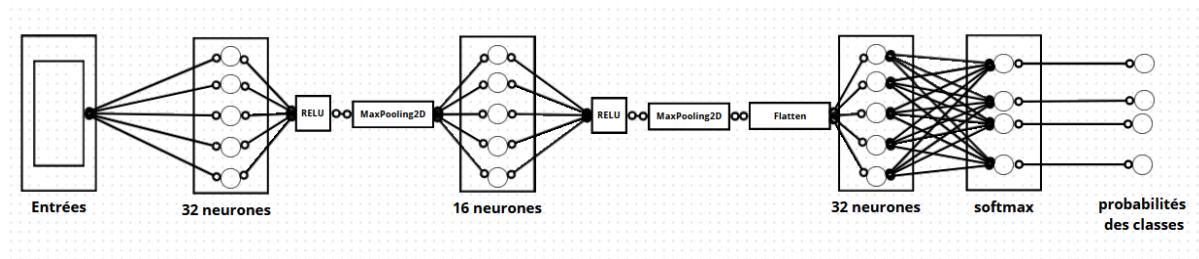


Capture n°9 - Résultat du filtrage des contours des images

Le filtrage à différentes valeurs de threshold n'étant pas satisfaisant et n'a aidant pas à augmenter l'efficacité du réseaux de neurones, nous avons donc commencé à voir quel autre type de filtre pourrait être plus adéquat. De plus, le nombre d'images à filtrer et à sauvegarder, consomme énormément de ressources et ne peut pas être exécutée sur notre version gratuite de Google Colab (RAM saturée). Nous nous sommes donc penchés sur un filtre GLCM qui permet de mettre en évidence les différences de textures et qui serait plus adéquat pour notre application, mais sans succès lors de l'implémentation en python.

d. Variations du réseau de neurones

Comme perspective d'amélioration, nous pouvons aussi nous appuyer sur l'amélioration de notre réseau de neurones. Nous sommes partis du principe que simplifier notre réseau de neurones et utiliser un meilleur "layers.Dropout" pourrait améliorer les performances de détection. Nous avons essayé une simplification du réseau de neurones et nous avons obtenu les résultats suivants.



Capture n°10 : Schéma du réseau de neurones simplifié

```

Le pourcentage d'erreur pour une tumeur glioma est : 38.56858846918489%
Le pourcentage de réussite pour une tumeur glioma est : 61.43141153081511%

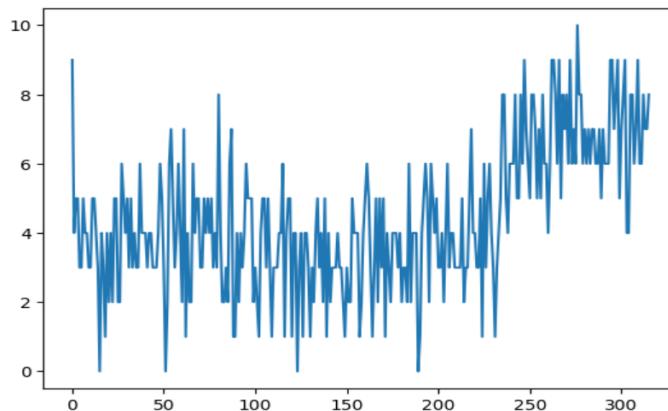
Le pourcentage d'erreur pour une l'absence de tumeur est : 82.72251308900523%
Le pourcentage de réussite pour une l'absence de tumeur est : 17.277486910994767%

Le pourcentage d'erreur pour une tumeur Menigioma est : 17.92336217552534%
Le pourcentage de réussite pour une tumeur Menigioma est : 82.07663782447466%

Le pourcentage d'erreur pour une tumeur Pituitary est : 57.572614107883815%
Le pourcentage de réussite pour une tumeur Pituitary est : 42.427385892116185%

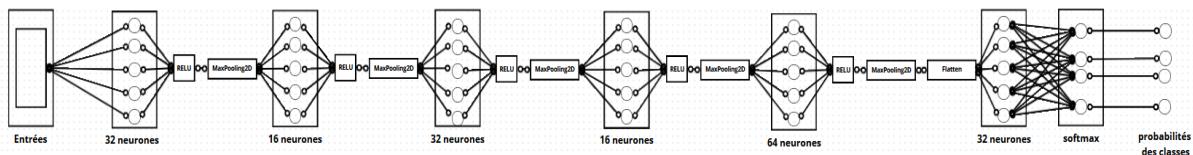
The medium error rate is : 49.19676946039982%
The medium success rate is : 50.89323053960018%

```



Capture n°11 : Résultats obtenus lors du test de notre second réseau de neurones

Nous pouvons constater qu'après une baisse de complexité de notre réseau de neurones et une meilleure gestion du surapprentissage, nous obtenons une baisse considérable de nos performances de détection. Nous sommes donc partis sur la complexification de notre réseau, afin de pouvoir constater si son amélioration pourrait améliorer nos performances. Nous avons fait comme hypothèse, qu'une complexification allait permettre une meilleure détection de chacune de nos classes.



Capture n°12 : Schéma du réseau de neurones complexifié

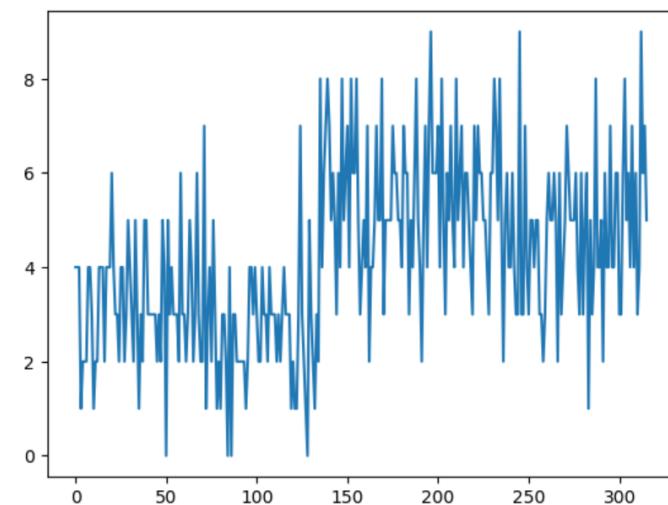
Le pourcentage d'erreur pour une tumeur glioma est : 28.82703777335984%
Le pourcentage de réussite pour une tumeur glioma est : 71.17296222664015%

Le pourcentage d'erreur pour une l'absence de tumeur est : 27.486910994764397%
Le pourcentage de réussite pour une l'absence de tumeur est : 72.5130890052356%

Le pourcentage d'erreur pour une tumeur Menigioma est : 68.4796044499382%
Le pourcentage de réussite pour une tumeur Menigioma est : 31.520395550061806%

Le pourcentage d'erreur pour une tumeur Pituitary est : 39.522821576763484%
Le pourcentage de réussite pour une tumeur Pituitary est : 60.477178423236516%

The medium error rate is : 41.07909369870648%
The medium success rate is : 58.92090630129352%



Capture n°13 : Résultats obtenus lors du test de notre troisième réseau de neurones

Comme finalité, nous avons pu constater que l'utilisation d'une nouvelle couche permet une même détection moyenne de nos classes et nous finissons par une détection moyenne de 58%, le même résultat pour une complexité plus importante. Nous avons quelques variations par rapport aux probabilités de chaque classe, mais le résultat final est identique. Nous pouvons donc constater que la complexification de notre réseau de neurones n'est pas utile. Nous sommes retournés sur notre premier réseau de neurones après l'ensemble des tests effectuées car c'est celui-ci qui offrait la qualité de détection la plus avantageuse avec une complexité moindre.

2. Perspectives d'améliorations

a. Délimiter la zone de la tumeur

Une manière de rendre le programme plus intéressant serait de s'appuyer sur des images d'IRM où les tumeurs sont entourées ou pointées par une flèche. N'ayant aucun jeu de données comportant ce genre d'information, nous avons dû laisser cette idée de côté. De plus, il est assez difficile de réaliser cette délimitation comme des tumeurs peuvent se généraliser dans l'ensemble du cerveau.

b. Utiliser un filtrage performant

Afin de faciliter l'étude de notre réseau de neurones, une perspective d'amélioration aurait été de mettre en place un filtrage auprès de nos images, comme dit précédemment le filtrage GLCM. Cela aurait permis à notre réseau de pouvoir analyser les tumeurs en fonction de l'étude de la texture des tumeurs.

Conclusion

Notre projet a ainsi connu plusieurs phases de développements et d'améliorations. Initialement axé sur la détection binaire de la présence ou de l'absence de tumeurs, nous avons élargi notre perspective en distinguant trois types de tumeurs spécifiques. Le projet a été structuré autour de recherches approfondies et de collecte de données de qualité sur Kaggle. Au fil des étapes, nous avons mesuré les performances de notre modèle, constatant des résultats satisfaisants pour certaines classes, mais des défis persistants, notamment dans la détection de l'absence de tumeur.

Dans une démarche d'amélioration, nous avons augmenté la taille de notre jeu de données, ce qui a eu des impacts variables sur la performance du modèle en fonction des classes. De plus, nous avons introduit des mécanismes pour éviter le surapprentissage, garantissant ainsi une meilleure généralisation du modèle. Malgré ces ajustements, nous avons identifié la nécessité de continuer à travailler sur des axes d'amélioration potentiels tels que le filtrage, notamment en explorant des techniques comme le GLCM pour analyser la texture des tumeurs.

Notre projet a abouti à un réseau de neurones capable de détecter les tumeurs avec une certaine précision. Cependant, la route vers l'amélioration continue, et des perspectives comme la délimitation précise de la zone de la tumeur et l'utilisation de filtres plus performants restent à explorer.

Dans l'ensemble, ce projet a été une opportunité enrichissante pour appliquer nos connaissances en apprentissage automatique à un domaine aussi crucial que l'imagerie médicale. Les enseignements tirés des défis que nous avons rencontrés nous seront précieux pour guider de futures améliorations et recherches dans le domaine de la détection et classification d'image par le biais de l'intelligence artificielle.

Annexe

Sources

*¹ : Accès au code Python sur Google Colab :

https://colab.research.google.com/drive/1l1f7zExY14tNPbj_yNCKH2QgnZ1sFK8-?usp=sharing

*² : Coder un réseau de neurones convolutif de classification d'image avec Python et Tensorflow :

<https://www.youtube.com/watch?v=6FHtTyZxS5s>

*³ : Détection du cancer du cerveau à l'aide d'un réseau de neurones :

<https://www.ijser.org/researchpaper/Detection-du-cancer-du-cerveau-a-l'aide-d-un-reseau-de-neurones.pdf>

*⁴ : Datasets utilisées lors de la première phase de développement de notre projet :

<https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset>

<https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri>

*⁵ : Datasets rajoutées lors de la seconde phase de développement de notre projet :

<https://www.kaggle.com/datasets/thomasdubail/brain-tumors-256x256>

<https://www.kaggle.com/datasets/denizkavi1/brain-tumor>

Informations

Nombre d'image d'entraînement lors de la première étape : 1176

Nombre d'image de test lors de la première étape : 529

Nombre d'image d'entraînement lors de la seconde étape : 4704

Nombre d'image de test lors de la seconde étape : 3161

Outils utilisés

Discord : <https://discord.com/>

Google Colaboratory : <https://colab.research.google.com/?hl=fr>

Google Drive : <https://www.google.com/intl/fr/drive/>

Librairie Tensorflow : <https://www.tensorflow.org/resources/libraries-extensions?hl=fr>

Tâches effectuées

Recherches documentaires : L'ensemble du groupe

Recherches des données d'images d'IRM : Nicolas, Quentin, Étienne

Mise en place du réseau de neurones convolutifs : Étienne, Nicolas

Étude et développement du filtrage : Lucas, Quentin

Script python : L'ensemble du groupe

Rédaction et lecture du rapport : L'ensemble du groupe

Réalisation de l'oral et réalisation du diaporama : L'ensemble du groupe

Lexiques

Dataset : Plusieurs données ayant un lien cohérent entre elles

GLCM : Filtrage qui s'appuie sur l'analyse de texture