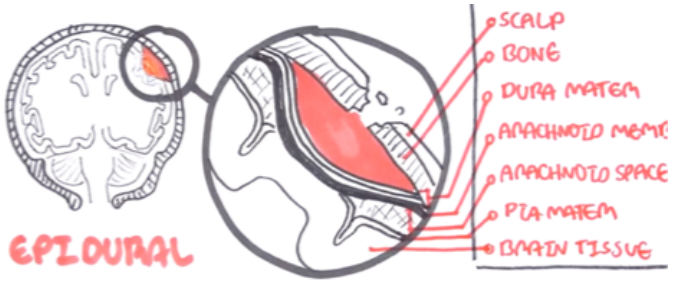
Introduction et contexte

Notre problématique porte sur la détection des hémorragies intracrâniennes chez l'Homme, ce projet est issu de la compétition Kaggle *RSNA Intracranial Hemorrhage Detection – Identify acute intracranial hemorrhage and its subtypes*, qui a débuté en septembre 2019 et s'est clôturée en novembre de la même année.

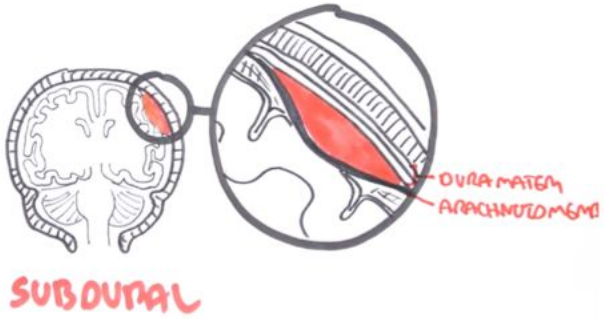
Une hémorragie intracrânienne correspond à un saignement localisé à l'intérieur de la boîte crânienne. Nous distinguons 5 sous-types d'hémorragies :

1. Hémorragies extra-axiales : occurrent à l'intérieur du crâne mais en dehors des tissus cérébraux :

   ➢ Épidurale : saignement entre le crâne et la membrane épidurale



   ➢ Subdurale : saignement entre la membrane dura et la membrane arachnoïde

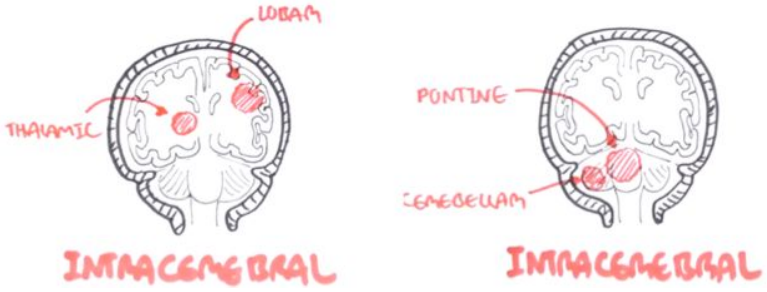


   ➢ Subarachnoïde : saignement dans l'espace subarachnoïde

2. Hémorragies intra-axiales : occurrent à l'intérieur des tissus cérébraux :

➢ Intraventriculaire : saignement à l'intérieur du système ventriculaire

➢ Intraparenchymateuse : saignement à l'intérieur des tissus



Les hémorragies intracrâniennes résultent dans la plupart des cas du dommage d'un vaisseau sanguin, dont les causes peuvent être physiques ou non traumatiques. La détection rapide et efficace de l'hémorragie est un enjeu crucial en médecine, il faut noter de plus qu'il s'agit d'un accident relativement commun (10 à 15% des accidents cérébraux) ce qui appuie

d’autant plus l’importance de sa détection. Les objectifs fixés pour ce projet sont ainsi la détection, la localisation et l’identification du type d’hémorragie mise en jeu.

Le dataset mis à disposition a été fourni par la Radiological Society of North America (RSNA), il est constitué de 674 258 données d’entrainement et de 78 545 données de test. Les données se présentent sous la forme d’images 2D de coupes de cerveaux -CT scans-, avec une image par patient (identifié par un ID).

Classiquement, c’est une méthode de détection par CT scan qui est employée pour repérer avec précision la présence de l’hémorragie, et le médecin exploite la localisation, taille et proximité de l’anomalie présente avec les structures environnantes pour poser son diagnostic. Le CT scan, ou Computerized Tomography scan, est une méthode d’imagerie médicale qui consiste à reconstituer les images 2D ou 3D de structures biologiques en se basant sur les mesures d’absorption des rayons X par ces tissu, via un traitement informatique. En pratique le patient est soumis au balayage d’un faisceau de rayons X, c’est l’analyse tomographique, dont l’absorption sous différents angles est relevée, c’est ensuite par traitement informatique que les images en coupe sont reconstituées (des "tranches" virtuelles de zones spécifiques).

Ce rapport vise à exposer la démarche générale suivie pour traiter la problématique posée, ainsi que les résultats obtenus. Tout d’abord, le jeu de données sera présenté -et est disponible dans le fichier *rapport.ipynb*-, ensuite l’exploration qui a été faite sur les notebooks de la compétition sera détaillée. Enfin, afin de proposer une méthode de détection satisfaisante, nous avons travaillé sur le développement d’un réseau de neurones convolutifs, réalisé dans le fichier *modelV4.ipynb*, dont les choix d’architecture et de paramétrage seront abordés, puis les résultats discutés.

Exploration des données

Les données sont des fichiers de type dicom, il s'agit d'images 2D de coupes de cerveaux, et par patient il y a une seule coupe du issue du CT scan (ou IRM). Les images sont labellisés selon 6 classes :

- any s'il y a au moins 1 hémorragie
- Et ensuite les 5 labels restants correspondant au type de l'hémorragie (épidural, subdural, etc)

Un fichier dicom contient les informations suivantes, ainsi présentées :

```
(0008, 0018) SOP Instance UID                   UI: ID_5c8b5d701
(0008, 0060) Modality                           CS: 'CT'
(0010, 0020) Patient ID                         LO: 'ID_7fa2490c'
(0020, 000d) Study Instance UID                 UI: ID_9f8138efb6
(0020, 000e) Series Instance UID                UI: ID_34c0b81444
(0020, 0010) Study ID                           SH: ''
(0020, 0032) Image Position (Patient)           DS: ['-112', '-20.596', '145.67']
(0020, 0037) Image Orientation (Patient)        DS: ['1', '0', '0', '0', '1', '0']
(0028, 0002) Samples per Pixel                  US: 1
(0028, 0004) Photometric Interpretation         CS: 'MONOCHROME2'
(0028, 0010) Rows                               US: 512
(0028, 0011) Columns                            US: 512
(0028, 0030) Pixel Spacing                      DS: ['0.4375', '0.4375']
(0028, 0100) Bits Allocated                     US: 16
(0028, 0101) Bits Stored                        US: 12
(0028, 0102) High Bit                           US: 11
(0028, 0103) Pixel Representation               US: 0
(0028, 1050) Window Center                      DS: ['40', '40']
(0028, 1051) Window Width                       DS: ['80', '80']
(0028, 1052) Rescale Intercept                  DS: "-1024"
(0028, 1053) Rescale Slope                      DS: "1"
(7fe0, 0010) Pixel Data                         OW: Array of 524288 elements
```

Figure 1 - contenu du fichier Dicom du patient 7fa2490c

Le dataset contient 674 258 patients dans les données d'entraînement et 78 545 dans les données de test mais la répartition entre les différents types d'hémorragies est inégale, comme nous pouvons l'observer :

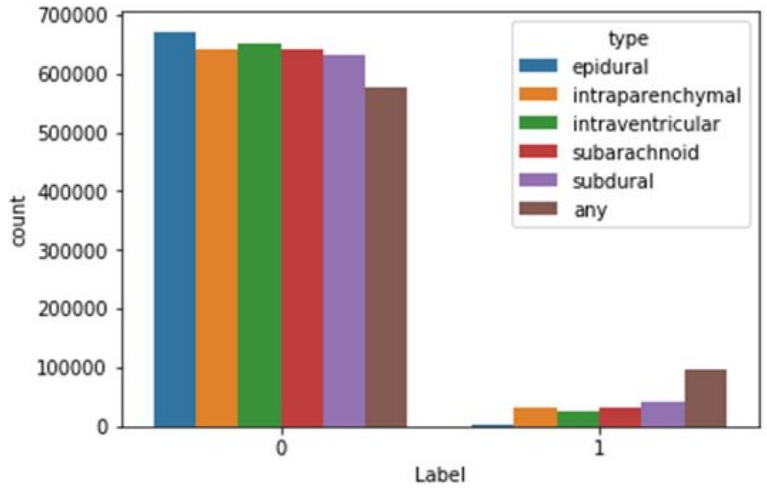


Figure 2 - répartition des labels suivant les types d'hémorragies

L'hémorragie de type épidural est nettement sous représentée, avec 2 761 hémorragies de ce type dans le dataset, ce qui peut s'avérer problématique car l'apprentissage du réseau de neurones ne peut se faire correctement ou être efficace avec un nombre trop faible de données.

De plus, il faut noter que certains patients peuvent présenter plusieurs types d'hémorragie, jusqu'à 5 :

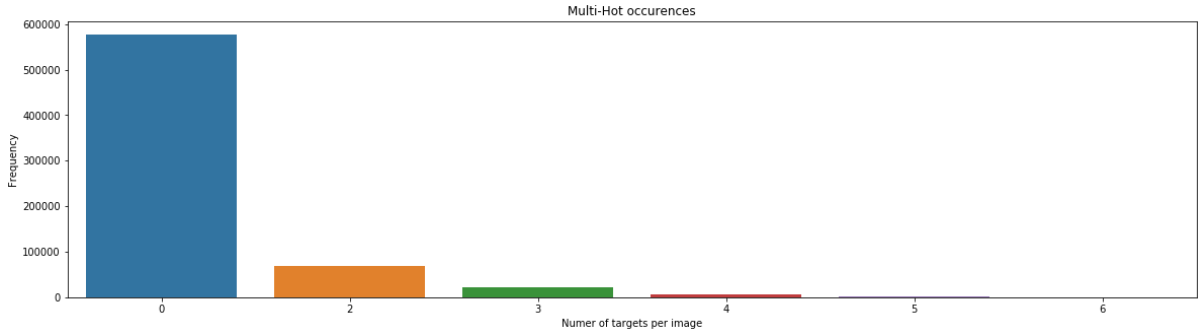


Figure 3 - nombre d'occurrence d'hémorragie pour un patient

Pour visualiser les CT scans proprement, on a besoin d'appliquer une transformation linéaire dépendant de « Rescale Intercept » contenu dans le fichier dcm et d'ensuite les centrer.

La valeur des pixels balayés par les fichiers dicom :

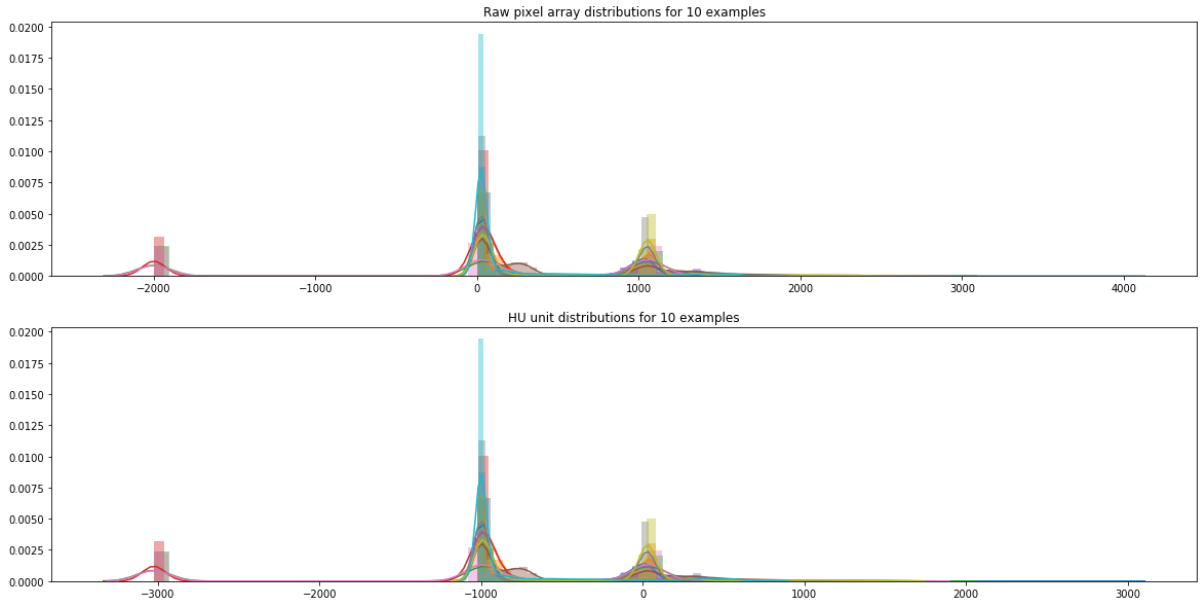
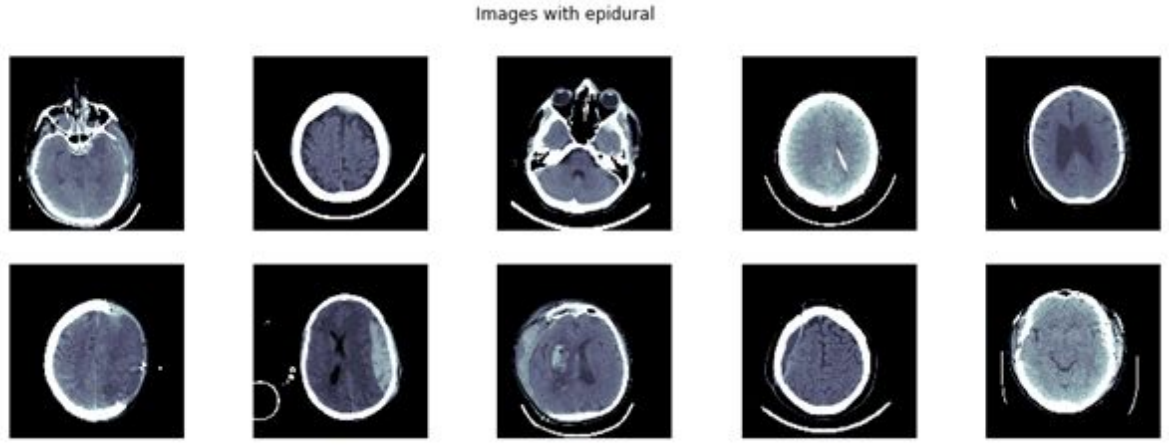
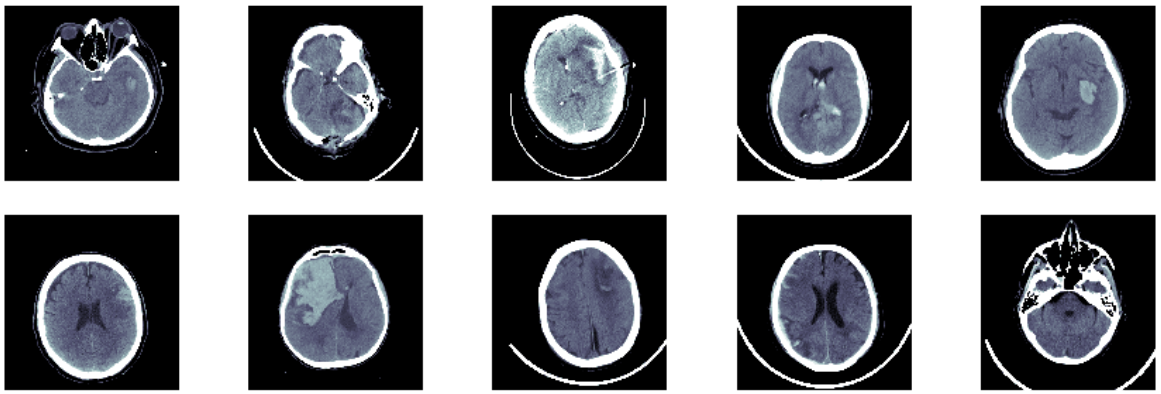


Figure 4 - répartition des pixels

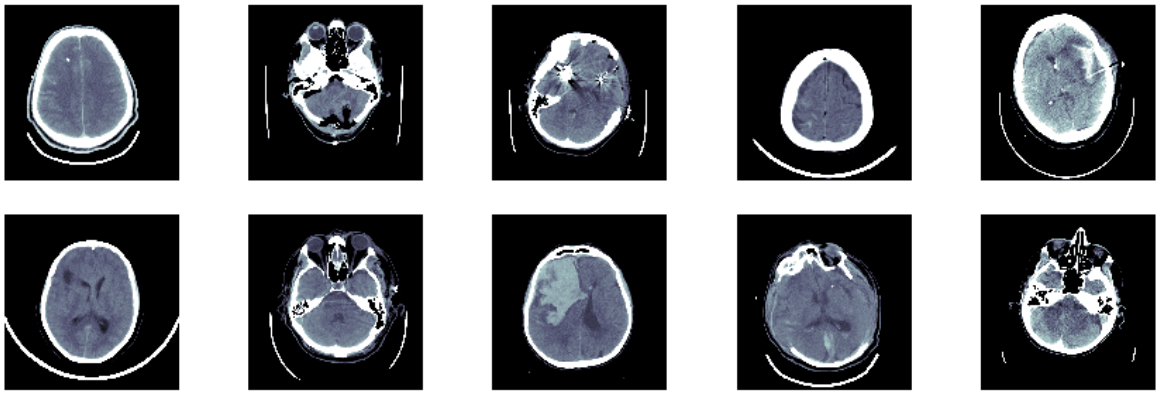
Voici les images après traitement, suivant les différents types :



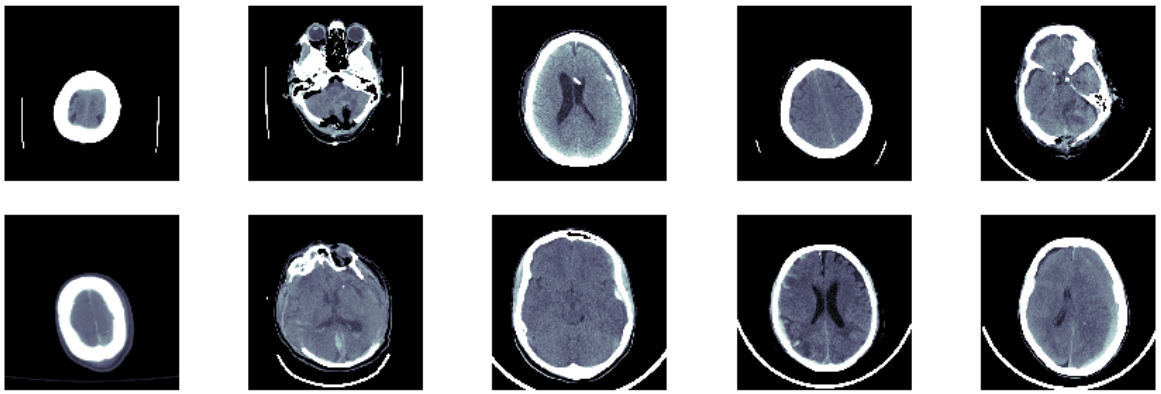
Images with intraparenchymal



Images with subarachnoid



Images with subdural



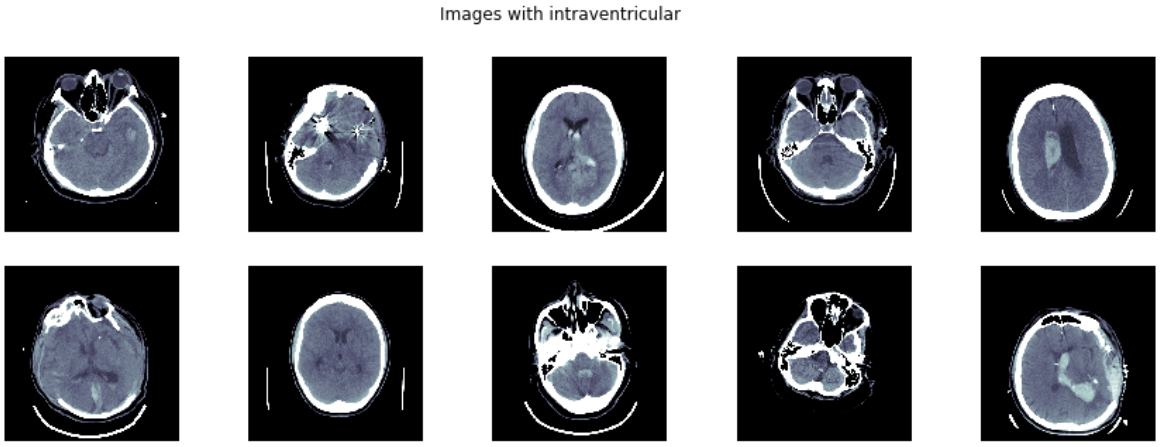


Figure 5 - visualisation de CT scans selon le type d’hémorragie

Exploration des notebooks

Il fut pertinent de commencer par explorer les notebooks déjà réalisées, sur le site de la compétition Kaggle, pour davantage comprendre et appréhender la problématique.

L'un des premiers objectifs abordé est le formatage les données du fichier type csv pour obtenir une ligne par patient avec 6 colonnes pour chaque type.

Soit le passage de :

	ID	Label
0	ID_63eb1e259_epidural	0
1	ID_63eb1e259_intraparenchymal	0
2	ID_63eb1e259_intraventricular	0
3	ID_63eb1e259_subarachnoid	0
4	ID_63eb1e259_subdural	0

A un tel tableau :

	Label					
Diagnosis	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
Image						
ID_000039fa0	0	0	0	0	0	0
ID_00005679d	0	0	0	0	0	0
ID_00008ce3c	0	0	0	0	0	0

Figure 6 - labels après formatage

Un tel format sera plus adapté pour le réseau par la suite, et c'est aussi ce qui est fait dans tous les notebooks présentés sur le site de la compétition.
Dans ces notebooks de plus, une modification est appliquée aux images afin de supprimer les informations non pertinentes (par exemple les tissus et les os) et souligner les informations utiles, en augmentant par exemple le contraste et en recadrant toutes les images, comme vu précédemment.

Néanmoins, l'exploration des notebooks disponibles s'est révélée peu fructueuse :

- L'un des premiers notebooks que nous avons exploré utilise un réseau inceptionV3 de keras sauf que nous avons eu plusieurs erreurs que personne n'a su résoudre donc

nous avons dû abandonner ce notebook.
https://www.kaggle.com/akensert/inceptionv3-prev-resnet50-keras-baseline-model

Par la suite nous avons décidé de créer un environnement python personnalisé car on ne pouvait pas retracer les environnements par défaut du GPU utilisés dans les notebooks pour pouvoir exécuter nos scripts pendant plusieurs heures sans interruptions.

- ➢ Pour un second notebook, il fallait installer Nvidia Apex, une extension de Pytorch, dont l'installation n'a pas fonctionné.
  https://www.kaggle.com/braquino/pytorch-resnext-32x8d-centercrop

- ➢ Sur d'autres notebooks, il y a eu des erreurs lorsque nous avons voulu entraîner le modèle, nous obligeant à abandonner juste avant d'avoir des résultats, ce qui fut très frustrant.
  https://www.kaggle.com/rsmits/keras-efficientnet-b2-starter-code
  https://www.kaggle.com/jesucristo/rsna-introduction-eda-models
  https://www.kaggle.com/allunia/rsna-ih-detection-baseline

En conclusion, nous avons eu énormément de problèmes avec les notebooks, mais aussi avec le GPU qui était très lent et s'éteignait parfois pendant de longues durées. Nous avons donc préféré créer notre propre réseau de neurones, notamment car la plupart des codes disponibles étaient peu accessibles à comprendre.

Développement d’un réseau de neurones convolutifs

Ce projet a consisté à construire un réseau de neurones convolutifs -travail réalisé par Marjolaine en totalité et exclusivement-, dans le but de classifier des images d'IRM selon les tumeurs des patients. Cette partie vise à présenter la démarche générale qui a été suivie pour traiter le problème, ainsi que les résultats obtenus.
Tout d'abord, des notions générales sur les réseaux de neurones convolutifs seront introduites, celles-ci serviront pour la suite du rapport. Dans un second temps, le jeu de donné sera présenté, ainsi que les étapes de pré-traitement de celui-ci pour le présenter au réseau. Ensuite, les choix d'architecture et de paramètres utilisés pour bâtir le modèle seront abordés. Enfin, les résultats obtenus seront exposés et discutés.

Notions générales sur les réseaux de neurons convolutifs

- *Buts et définitions*

Le problème considéré dans le cadre de ce projet appartient à la classification d'images. L'objectif est de savoir reconnaître la présence d'un objet cible, parmi différentes classes d'objet possibles, dans une image donnée. Dans le cas le plus simple, la classification est binaire : pour chaque image, il s'agit de discriminer si l'objet est présent ou absent. Le problème peut s'étendre à une classification multiple, lorsque plusieurs objets peuvent être présents dans les images. Lorsque chaque image ne peut appartenir qu'à une seule classe, on parle de "multi-class classification". Lorsqu'à l'inverse, chaque image peut contenir plusieurs objets, on parle de "multi-label classification".
Dans le cas présent, il s'agit d'identifier la présence de tumeurs cérébrales dans des images d'IRM ou CT scans, et plus précisément, de déterminer le type de tumeur parmi cinq profils déjà présentés (épidural, intraparenchymal, intraventriculaire, subarachnoïde, subdural).

Pour aborder ce problème, l'approche choisie consiste à bâtir un réseau de neurones. Formellement, un réseau de neurones peut être défini comme une fonction algébrique, dont la variable d'entrée est une image, et dont le résultat est la (ou les) classe(s) auxquelles cette image appartient. Les différentes dimensions de la variable d'entrée contribuent au calcul de la prédiction. Dans le cas des images, ces dimensions sont les intensités de couleurs en chaque pixel. Le résultat calculé par la fonction dépend de paramètres.

L'enjeu est de trouver l'architecture du réseau ("forme" de la fonction) et les paramètres optimaux, qui permettent de prédire avec le plus de précision possible la classe à laquelle appartient chaque image.

Il existe différentes architectures de réseaux de neurones. De manière générale, un réseau peut être vu comme un graphe, comprenant un ensemble d'unités ("neurones") connectés les uns aux autres, organisés en plusieurs couches. Les connections entre les unités, appelées poids, s'établissent entre couches, et font partie des paramètres à optimiser. C'est le profil de connections entre couches qui détermine le type d'architecture, représentant la manière dont les neurones s'influencent les uns les autres.
A chaque présentation d'image, chaque neurone est dispose d'une valeur d'activation. Celle-ci dépend d'un input, qui est la somme des activations des neurones auxquels il est lié, pondérée par les poids. L'activation dépend également d'une fonction non-linéaire appliquée à l'input. Différents choix sont possibles pour les fonctions d'activation, et seront discutées plus loin.

- ➢ *Spécificité du modèle convolutif*

Dans le cas le plus classique, un réseau de neurones peut être constitué exclusivement de couches denses. Dans ce cas, chaque neurone d'une couche est lié à chaque neurone de la couche suivante. La dernière couche du réseau contient autant d'unités que de classes à prédire. Chacune de ces unités renvoie la probabilité que l'image appartienne à la classe représentée.
Toutefois, d'autres architectures sont envisageables, et peuvent s'avérer plus adaptées pour certains types de problèmes. L'idée est de trouver une architecture qui exploite une régularité spécifique au type de données considérées.
Les réseaux de neurones convolutifs correspondent à une architecture particulièrement adaptée aux problèmes de classification d'images. En effet, leur architecture tire parti de la corrélation spatiale entre les pixels, en traitant des zones de l'espace progressivement, du plus local au plus large.

Cette architecture a été inspirée de l'organisation des systèmes visuels biologiques. Les neurones réels sont caractérisés par des champs récepteurs, aires de la scène visuelle modifiant la réponse de ce neurone quand un stimulus approprié y survient.
Dans les premières couches, les neurones sont sensibles à des aspects simples des stimulus, tels que l'intensité lumineuse, en une zone restreinte de la scène visuelle. En progressant dans la hiérarchie des couches, les champs récepteurs convergent et s'élargissent, définissant des caractéristiques plus complexes auxquelles répondent les neurones. Par exemple, les poids entre un neurone d'une couche i et un ensemble de neurones de la couche i-1 peuvent lui permettre de détecter des bords (par changement de contraste), dans une orientation particulière. Plus en aval, des angles pourront être reconnus en combinant plusieurs orientations de bords, et ainsi de suite pour la reconnaissance de formes. Finalement, cette

organisation confère aux neurones une faculté de détection invariante par rapport à la position, à l'orientation, jusqu'à leur permettre de reconnaître spécifiquement un type d'objets.

Les réseaux de neurones convolutifs sont caractérisés par des filtres (matrices de poids), qui sont les analogues des champs récepteurs. Une couche contient un ensemble de filtres, chacun pouvant être interprété comme une caractéristique de l'image susceptible d'être présente en de nombreux endroits, telle qu'une bordure, une forme, etc. Au niveau de la première couche, chaque filtre est convolué en chaque point de l'image. Ce calcul génère une valeur reflétant la présence de la caractéristique que le filtre représente dans la zone de l'image centrée sur ce point. En réitérant cette opération en tous les points de l'image, une "carte" de la caractéristique est créée. Les cartes issues des différents filtres de la couche sont ensuite convoluées de même avec les filtres de la couche supérieure, et ainsi de suite.
Entre chaque couche, un regroupement, appelé pooling, peut s'effectuer entre les valeurs obtenues en des points voisins. Généralement, seule la valeur maximale est conservée parmi un ensemble de points ("maxpooling"). Ceci a pour effet de réduire la dimensionnalité et le nombre de paramètres, mais permet également d'accroître l'invariance à la position. Par exemple, il y a huit directions dans lesquelles l'image d'entrée peut être translatée par un seul pixel. Si le pooling est effectué sur une région de dimension 2x2, trois de ces huit configurations possibles produiront exactement la même sortie au niveau de la couche supérieure.
A l'issue de ce traitement hiérarchique des données, une série de couches denses exploite l'information fournie par la dernière couche convolutive, comme exposé précédemment.

- ➢ *Méthode d'optimisation des poids*

L'optimisation des paramètres est réalisé au cours d'une phase d'apprentissage : les paramètres sont modulés par itération, jusqu'à obtenir des performances optimales sur un jeu de données. Dans le cadre de l'apprentissage supervisé, le réseau est confronté à des images dont la classe a été déterminée au préalable. Ceci permet, à chaque présentation, de calculer l'erreur de prédiction commise par le réseau. La valeur de cette erreur dicte la manière dont les coefficients sont modifiés pour l'étape suivante.
Plus précisément, l'optimisation des coefficients revient à minimiser une fonction de coût, quantifiant l'erreur commise par le réseau sur l'ensemble du jeu de données. La méthode de descente de gradient indique la magnitude et le sens dans lequel chaque paramètre doit être mis à jour. La définition d'une fonction de perte dépend du type de classification envisagé, mais aussi du type de problème. Ces considérations seront développées dans la suite du rapport.

Un des écueils à éviter lors de l'optimisation est le phénomène de sur-apprentissage (overfitting). Celui-ci peut survenir lorsque le réseau est entraîné trop longtemps sur un même jeu de données : les paramètres s'adaptent de manière trop étroite aux caractéristiques du jeu

de données. Ceci s'accompagne d'une perte de généralité, c'est-à-dire d'une diminution du pouvoir prédictif lorsque le réseau est confronté à des jeux de données nouveaux.
Pour contourner ce risque, l'apprentissage peut suivre une procédure de validation croisée. Le jeu de données est scindé en deux parties, celui d'entrainement, et celui de validation, sur lesquels le coût est calculé séparément. Seule la partie d'entraînement est utilisée pour mettre à jour les paramètres, tandis que la partie de validation permet de vérifier que le coût continue à diminuer sur un jeu de données n'ayant pas servi à la mise à jour. Cette approche fournit un critère d'arrêt de l'apprentissage : lorsque le coût commence à augmenter sur la partie "test" du jeu de données.

Jeu de données et étapes de pré-traitement

➢ *Choix de la stratégie*

Comme présenté précédemment, le jeu de données est un ensemble d'images d'IRM en niveau de gris. L'objectif est une classification multiple : pour chaque image, il s'agit de déterminer le type de tumeur parmi cinq profils (épidural, intraparenchymal, intraventriculaire, subarachnoïde, subdural), ou si le patient ne présente aucune tumeur (healthy). Le modèle contient donc six classes.
Une rapide exploration des données montre que chaque patient peut être affecté de plusieurs types de tumeurs simultanément. Ce constat conduit à adopter une approche de "multilabel classification" plutôt qu'une approche de "multi-class classification"..

Pour traiter ce problème, plusieurs stratégies peuvent être envisagées.
La première stratégie, "One vs. Rest", consisterait à construire cinq modèles de classification binaire, pour reconnaître chacune des tumeurs indépendamment. Néanmoins, une analyse des corrélations entre tumeurs suggère que certaines tumeurs sont plus susceptibles d'être associées que d'autres. Ces corrélations pourraient constituer une information intéressante que le réseau pourrait exploiter pour augmenter la précision des prédictions. Cette idée mène à privilégier la construction d'un unique réseau, prenant en compte toutes les possibilités simultanément.

➢ *Pre-processing*

Avant d'être soumis au réseau, le jeu de données doit être préparé dans un format approprié.

Une première étape consiste construire un vecteur de "labels", indiquant pour chaque image la classe à laquelle celle-ci appartient. Pour cela, on procède à un encodage one-hot : à chaque image est associé un vecteur binaire de 6 éléments, chacun représentant une classe.

L'élément associé à une classe donnée contient 1 si l'image appartient à cette classe, 0 sinon. Les labels ont été extraits d'un fichier csv, et formatés dans un tableau dont chaque colonne représente une classe. Le fichier csv contenait six lignes pour chaque patient. Les cinq premières indiquaient la présence/absence de chaque tumeur, tandis que la dernière ('any') rapportait la présence d'une tumeur, quelle qu'elle fût. Cette dernière information a été remplacée par une colonne 'healthy', correspondant à la classe des individus sains.
Dans un second temps, il est nécessaire de vérifier la cohérence des données. Ceci implique notamment de s'assurer de l'absence de données manquantes, et de vérifier que toutes les images présentent les mêmes dimensions. Les éléments ne remplissant pas ce dernier critère ont été exclus du jeu de données.

Dans un troisième temps, il s'agit de répartir les images en deux jeux de données, train et test. Le jeu de données train sert à l'apprentissage, tandis que le jeu de données test est réservé pour évaluer les performances du réseau à l'issue de l'apprentissage. On choisit généralement d'attribuer 80% des images au jeu de données train.
Pour construire un jeu de données, il a semblé important de s'assurer que les différentes classes soient également représentées dans les données. En effet, la classe 'healthy' est beaucoup plus fréquente dans l'ensemble du jeu de données. Un échantillonnage sélectif a donc été réalisé, afin d'obtenir une distribution approximativement uniforme des tumeurs dans le jeu de données.
Il a enfin fallu regrouper les images dans un tableau multidimensionnel, et ordonner les labels de sorte à assurer la correspondance dans chaque couple image-label présenté au réseau à chaque itération.

Dans certains cas, il peut être utile de normaliser les valeurs d'intensité dans chaque image.

Construction du modèle

➢ *Choix des hyperparamètres du réseau*

Une série d'hyperparamètres doivent être spécifiés pour les différentes couches du réseau.

Paramètres des couches convolutives :

--- Nombre de filtres ---

Pour chaque couche, le nombre de filtres représente le nombre de caractéristiques à rechercher dans la "carte" issue de la couche précédente. Ce nombre de filtres est choisi assez restreint pour les premières couches. En effet, à une échelle très locale dans l'image, la diversité des agencements d'intensités doit être faible. Par exemple, huit filtres ont été choisis pour la première couche, en se fiant à l'idée de huit directions spatiales. Au fil de la progression dans le réseau, le nombre de filtres peut s'élargir, puisque les couches

représentent des objets de plus en plus abstraits, dont la diversité est plus grande. Par exemple, les deux dernières couches possèdent 16 et 32 filtres respectivement.
Toutefois, le nombre de filtres a été limité pour des raisons computationnelles, concernant le nombre de paramètres à optimiser. L'enjeu est également de conserver un niveau de simplicité dans le modèle qui limite le risque de sur-apprentissage.

--- Dimension d'un filtre ---

Les filtres de petite dimension ont été privilégiés. En effet, considérons deux options pour réduire la dimension de l'image et agréger l'information. Une même réduction peut être obtenue soit par combinaisons de plusieurs petits filtres (par exemple, deux filtres 3x3 pour réduire l'image par 4), soit par un seul grand filtre (par exemple, un filtre 5x5). Or, les petits filtres offrent une meilleure extraction d'information pertinente. Ils représentent des champs récepteurs étroits, donc des caractéristiques précises. A l'inverse, un filtre large correspond à une information plus diffuse spatialement, et prend donc en compte des caractéristiques plus générales des images, contenant moins d'information. Les petits filtres mettent en exergue des effets non linéaires dans l'image, tandis qu'un filtre large tend vers un calcul plus linéaire.
D'autre part, on opte en général pour des filtres de dimension impaire. Ceci permet de limiter les effets de distorsion entre couches. Considérons par exemple unique un pixel d'intensité positive entouré de pixels d'intensité nulle. Un filtre de dimension impaire générera dans la "carte" supérieure un pic d'intensité centré sur ce point. En revanche, un filtre de dimension paire ne conserverait pas la localisation du pic.
Ces considérations ont conduit à choisir des filtres de dimension 3x3 (un filtre de dimension 1x1 étant exclu, car ne permettant pas d'agrégation).

--- Padding ---

Lors d'une convolution, un filtre est déplacé successivement en toutes les positions possibles. Si l'image est l'image initiale, aucun filtre ne pourra être centré sur les pixels les plus externes. Ceci conduit à réaliser un nombre de convolutions inférieur au nombre de pixels, et donc à perdre de l'information en bordure de l'image.
Pour éviter ce phénomène, le padding consiste à entourer l'image de pixels contenant des valeurs nulles, de sorte à pouvoir définir une convolution même aux pixels externes. De cette manière, la dimension de la "carte" obtenue après convolution est identique à celle de l'image sur laquelle elle a été calculée.
Pour la première couche, il n'a été jugé utile d'ajouter un padding. En effet, l'examination des images montre que les bordures correspondent au milieu autour du scanner, et ne contiennent donc pas d'information susceptible d'être reliée aux tumeurs. En revanche, un padding a été ajouté pour toutes les autres couches, car l'information commence a être agrégée et ne devrait pas être perdue.

--- Strides ---

Ce paramètre correspond au nombre de pixels dont le filtre est décalé entre deux convolutions. Lorsque ce paramètre vaut 1, le filtre est appliqué en chaque point de l'image. Lorsqu'il vaut 2, le filtre omet un pixel sur deux lorsqu'il parcourt une ligne ou une colonne.
Choisir une valeur supérieure à 1 permet de réduire la dimension de l'image, et par conséquent le nombre de paramètres à optimiser. Mais elle réduit en contrepartie la précision de l'information collectée. C'est pourquoi ce paramètre est généralement maintenu à 1, sauf

éventuellement au niveau des premières couches, où l'information est plus susceptible d'être redondante. Dans le cas présent, comme les images du jeu de données sont de grande dimension (512x512), une valeur de 2 a été considérée pour la première couche, et comparée à une valeur de 1. Il s'est avéré que le réseau présentait de meilleurs performances pour une valeur de 1, c'est donc celle qui a été conservée.

Paramètres des couches max-pooling :

--- Taille du regroupement ---

Le nombre d'unités synthétisés dans un regroupement revient à définir une taille de matrice. Le choix a été de regrouper les éléments par 4, avec une matrice de taille 2x2. Ceci diminue la dimension par 2 à la fois dans la hauteur et la largeur de l'image, et donc à ne conserver que les 25% des activations maximales.
La taille du regroupement a été limité, car un regroupement trop poussé pourrait conduire à perdre trop d'information.

Paramètres des couches denses :

--- Nombre d'unités ---

Après la dernière couche convolutive, deux couches denses ont été ajoutées.
Pour la première couche dense, un compromis a dû être trouvé entre un nombre d'unités suffisant pour une classification précise, et le nombre de paramètres à optimiser. En effet, chaque unité est connectée à l'ensemble des unités de la dernière couche convolutive. Différentes valeurs ont été testées successivement, et un nombre de 200 unités a été retenu.
La seconde couche dense possède 6 unités, puisque chacune doit fournir la probabilité que l'image appartienne à l'une des classe.

Paramètres généraux des couches :

--- Fonction d'activation ---

La fonction d'activation définit la magnitude avec laquelle un neurone répond aux inputs pondérés issus des neurones auxquels il est lié.
Ces fonctions présentent la propriété d'être non-linéaires. En effet, une fonction linéaire est moins intéressante pour repérer des caractéristiques complexes des données. Ce fait peut être appréhendé en imaginant un réseau constitué uniquement de fonctions d'activation linéaires. Un tel réseau serait équivalent à une unique couche linéaire, par combinaison. Il correspondrait dont en fait à un modèle de régression linéaire, donc le pouvoir prédictif est limité, car seule une relation linéaire entre les données peut être décelée. D'autre part, une fonction linéaire pose problème pour l'apprentissage. La dérivée d'une telle fonction est constante, donc sans lien avec l'input. Ceci empêche d'appliquer l'algorithme de back-propagation, pour remonter aux coefficients de l'input qui fournissent les meilleures prédictions.
Parmi les fonctions non-linéaires, plusieurs alternatives peuvent être envisagées. La fonction utilisée entre les couches intermédiaires est la fonction ReLU (Rectified Linear Unit). Elle comporte une partie constante (nulle) pour tout input négatif, et une partie linéaire en réponse aux inputs positifs. Un avantage de cette fonction est qu'elle n'est pas saturante. Cette caractéristique importe pour la back-propagation : une fonction saturante a une sensibilité limitée à l'input, car une fois saturée, la fonction prend la même valeur quel que soit l'input.

Ceci freine l'apprentissage, puisqu'il devient difficile de déterminer dans quelle direction modifier les paramètres pour améliorer le coût.
Pour la couche finale, à l'inverse, une fonction saturante est nécessaire. En effet, l'objectif est que le réseau calcule la probabilité que l'image appartienne à chaque classe. Ceci implique que la valeur calculée par le réseau soit comprise entre 0 et 1. Dans le cas d'une classification "multi-classes", les probabilités fournies par les unités de la dernière couche doivent sommer à 1, car l'image ne peut appartenir qu'à une seule classe. Dans ce cas, la fonction Softmax ext utilisée. En revanche, dans le cadre d'une classification "multi-labels", les probabilités doivent être indépendantes. On utilise donc dans ce cas une fonction sigmoïde.
Dans le cas du réseau considéré, la fonction ReLU a donc été sélectionnée pour les fonctions d'activation intermédiaires, tandis que la fonction sigmoïde a été retenue pour la dernière couche, dans le cadre d'une classification "multi-labels".

➢ *Recherche d'une architecture*

En plus des paramètres des couches, l'enjeu est d'identifier une architecture adaptée à la classification du jeu de données. Il s'agit de déterminer le nombre et le type de couches constituant le réseau, ainsi que leur agencement.

L'agencement des couches suit l'architecture classique mentionné dans le paragraphe introduisant les réseaux convolutionnels. Le principe est d'alterner les couches convolutives et les couches de maxpooling, jusqu'à déboucher sur des couches denses, comme évoqué précédemment.

La détermination du nombre de couches convolutives a été effectuée par essais-erreurs. En partant d'un réseau simple à une couche convolutive et deux couches denses, le nombre de couches convolutives a été augmenté progressivement. A chaque tentative, les paramètres des couches précédemment mentionnés étaient recherchés. Cette approche a été poursuivie de sorte à accroître les performances du réseau, tout en gardant trace de la meilleure architecture obtenue jusqu'alors. Là-encore, un compromis a dû être trouvé entre le nombre de paramètres, multiplié par l'approfondissement du réseau, et les performances.

D'autres types de couches ont été essayées, dans une optique de régularisation. L'objectif est de réduire le risque de sur-apprentissage par différents moyens.
La première stratégie est le dropout. A chaque étape d'apprentissage, un sous-ensemble d'unités est sélectionné aléatoirement dans une couche. Ces unités sont ignorées lors du calcul d'activations et lors de la back-propagation. D'une part, cette approche permet de limiter le sur-apprentissage. En effet, elle consiste à former de nombreux réseaux "différents" sur différentes parties des données, chacun étant choisi au hasard dans le réseau complet. Ainsi, si une partie du réseau devient trop sensible à un certain bruit dans les données, d'autres parties pourront compenser ce sur-apprentissage, car elles n'auront pas été confrontées aux échantillons contenant ce bruit. D'autre part, cette approche empêche les différentes unités du réseau de devenir trop corrélées dans leur activité. Par exemple, avec le dropout, si une unité apprend à reconnaître les orientations horizontales, une autre unité pourra se spécialiser davantage dans la détection des orientations les verticales.
La seconde méthode est celle de la batch normalisation. Elle consiste à redimensionner les valeurs produites par une couche, de sorte à ce que leur distribution soit centrée réduite dans

chaque batch d'apprentissage. La normalisation ajoute deux paramètres, afin de multiplier les valeurs issues de la couches par l'analogue d'un "écart-type", et de leur ajouter une "moyenne". La batch normalisation peut accroître la stabilité du réseau, en offrant une solution au problème de "covariate shift". Pendant l'apprentissage, à mesure que les paramètres des couches précédentes changent, la distribution des entrées de la couche actuelle change en conséquence. Ainsi, la couche actuelle devrait constamment se réadapter aux nouvelles distributions. Or, avec la batch-normalisation, lors de la back-propagation, il devient possible de ne modifiant que ces deux paramètres, au lieu de perdre la stabilité du réseau en changeant tous les paramètres.
La littérature scientifique en machine learning suggère que ces deux méthodes peuvent s'avérer incompatibles. C'est pourquoi il a été jugé plus prudent de les tester successivement. La couche de batch normalisation a été placée en sortie de la première couche convolutive, là où la variation est susceptible d'avoir le plus d'influence sur les couches en aval. Son effet s'est montré néfaste sur les performances, et, de façon assez prévisible, assez similaire à celle d'une normalisation des images. La couche Dropout, quant à elle, a été testée avant la couche dense. Elle s'est accompagnée d'une légère altération des performances, mais a été conservée dans l'optique de traiter des jeux de données plus vastes.

➢ *Hyperparamètres d'apprentissage*

Un dernier type de paramètres à considérer englobe les paramètres d'apprentissage et d'entraînement du réseau.

--- Epochs ---

Ce paramètre indique le nombre de fois où la totalité du jeu de données sera soumise au réseau. Un compromis doit être trouvé, de sorte à ce que le nombre d'epochs soit suffisant pour que des paramètres efficaces soient atteints, sans toutefois conduire au sur-apprentissage.
Pour identifier le nombre d'epochs optimal, on peut utiliser l'approche "early stopping". Elle consiste à mettre un terme à l'apprentissage automatiquement, dès qu'un critère est satisfait. Généralement, le critère porte sur le jeu de données de validation. Souvent, le critère est de cesser l'apprentissage dès que la fonction de coût commence à croître sur le jeu de données de validation, ou bien que la précision (accuracy) commence à décliner. Dans le cas présent, néanmoins, il a été jugé plus pertinent d'axer le critère sur l'évaluation du "recall", comme indiqué dans la rubrique "métrique".

--- Batch ---

La taille du batch correspond au nombre d'échantillons traités par le réseau avant la mise à jour de ses paramètres. Ce paramètre doit être inférieur ou égal au nombre d'échantillons dans le jeu de données d'apprentissage.
Généralement, la taille du batch est choisie inférieure au nombre d'échantillons dans le jeu de données. Une première raison est computationnelle : une taille de batch plus restreinte requiert moins de valeurs à engranger temporairement en mémoire avant la mise à jour des paramètres. Une seconde raison concerne la vitesse de convergence : le réseau est mis à jour avant même d'avoir été confronté à l'ensemble des données. Enfin, une troisième raison est

relative au pouvoir de généralisation. En effet, le réseau est confronté à des sous-échantillons, dans lesquels le bruit peut être plus important que dans l'ensemble des données. Ceci peut conduire à une dynamique d'ajustement des paramètres empêchant le réseau de sur-apprendre sur l'ensemble d'entraînement.
Différentes tailles de batch ont été testées, celle qui a été retenue est la valeur de 32.

--- Fonction de coût ---

La fonction de coût quantifie l'erreur commise par le réseau dans sa prédiction des classes. Pour un état donné du réseau, elle peut être calculée sur un ensemble d'échantillons (batch), ou bien sur l'ensemble du jeu de données (epoch). Cette fonction dicte la magnitude avec laquelle les paramètres doivent être modifiés à l'étape suivante.
Dans le cas d'une classification "multi-labels", on peut utiliser une fonction de coût identique au cas binaire : l'entropie croisée binaire (binary cross-entropy). En effet, le problème de classification "multi-labels" peut être vu comme autant de problèmes de classification binaires que le nombre de classes.
Cependant, il peut être envisagé d'ajuster la fonction de coût, de sorte à ce qu'elle favorise certains objectifs. Notamment, dans le cas considéré, les matrices de confusion ont permis de mettre en évidence que le réseau avait plus de mal à apprendre certaines classes. La fonction de coût a donc été pondérée, de sorte à ce que l'erreur de prédiction sur ces classes contribue davantage à l'erreur globale, et que les paramètres soient modifiés en conséquence. La définition d'une nouvelle fonction de coût, différente de l'entropie croisée, aurait également pu être considérée pour correspondre mieux aux objectifs de la détection de tumeurs (voir la rubrique suivante).

--- Métrique ---

Pour évaluer les performances du réseau, il est possible d'employer une métrique. La métrique peut être calculée à partir d'une matrice de confusion, comparant les prédictions fournies par le réseau aux classes. Cette matrice répartit les échantillons en quatre catégories : les vrais positifs (TP) sont les images dans lesquelles le réseau a identifié un type de tumeurs qui y est véritablement présent, les faux positifs sont les images dans lesquelles le réseau a identifié une tumeur qui n'y existe pas, les vrais négatifs sont les images dans lesquelles le réseau n'a pas identifié de tumeurs et qui n'en présentent effectivement pas, les faux négatifs sont les images dans lesquelles le réseau n'a pas identifié de tumeurs alors qu'une tumeur y est présente.
Le choix de la métrique dépend du problème posé. Dans le cas général, la métrique utilisée est l'accuracy, qui quantifie l'ensemble des précisions justes. Toutefois, l'objectif de la classification des tumeurs est plutôt de ne pas ignorer de tumeur, au risque de perdre en fiabilité en indiquant des fausses-alarmes. La métrique adaptée pour cela est le "recall", qui mesure la proportion de tumeurs trouvées par le réseau parmi l'ensemble des images contenant des tumeurs.

➢ *Résultats*

Pour des raisons de temps de calcul, le réseau a été entraîné sur une portion très limitée du jeu de données total, comprenant 25% de 3000 échantillons. La priorité était dans un premier

temps de construire une architecture par essais-erreurs, ce qui nécessitait de pouvoir réitérer la procédure d'entraînement à de nombreuses reprises. La difficulté résidait notamment dans le fait que la modification d'un hyperparamètre pouvait nécessiter de modifier tous les autres hyperparamètres en conséquence.
Pour étendre le réseau à des jeux de données plus vastes, il pourrait être intéressant de considérer des approches plus automatiques de recherche des hyperparamètres. Par exemple, la méthode cross-validation grid search scanne un espace de paramètres spécifié au préalable (grid), et conserve la meilleure combinaison de paramètres obtenus.

Le réseau a été entraîné sur 20 epochs, sans que le critère d'arrêt ne soit atteint. La métrique recall, bien que abordant la saturation, n'a jamais décliné sur le jeu de données de validation. L'évaluation sur un nouveau jeu de données donne une valeur de recall de 0.750. En revanche, le coût a entamé une croissance lente, à la limite de la stagnation, dès les premières epochs, ce qui manifesterait un sur-apprentissage précoce.
Ces deux résultats semblent paradoxaux, et soulèvent la question de la pertinence de poursuivre l'apprentissage au-delà de 20 epochs. Cette situation n'est d'ailleurs pas exactement équivalente au cas classique de sur-apprentissage, dans lequel le recall devrait décroître sur les données de validation. Une tentative d'explication pourrait s'appuyer sur le fait que la fonction de coût est définie sur les probabilités calculées par le réseau, tandis que le recall est défini sur la proportion de classes prédites, qui dépend du seuil de probabilité à partir duquel une image est attribuée à une classe. Une même classification peut donc être obtenue à partir de différentes probabilités calculées. Supposons que le réseau commence à affecter de grandes probabilités à certaines classes. Dans ce cas, une seule prédiction fausse conduit à une augmentation considérable du coût, alors qu'elle n'a que peu d'impact sur le recall. C'est pourquoi il pourrait être envisagé de redéfinir une fonction de coût permettant d'optimiser le recall.

Principes mathématiques d'image processing

Par manque de temps, nous n'avons pas pu réaliser de preprocessing d'images plus poussé, mais nous présentons comme ouverture les recherches effectuées dans cette optique.

Une image 2D peut être représentée par une fonction f -fonction image- de R^2 :
$(x, y) \mapsto f(x, y)$ où $f(x, y)$ est non nulle et finie et définie par :

1. La quantité d'illumination de la source incidente sur les objets $i(x, y)$ qui dépend de la source
2. La quantité d'illumination transmises (car les images ont été obtenues par traitement impliquant des rayons X) par les objets $0 < r(x, y) < 1$ qui dépend des caractéristiques des objets images

$\Rightarrow L_{min} < f(x, y) < L_{max}$ où $f(x, y)$ correspond au niveau de gris au point (x, y). Dans notre cas nous sommes ramenés à l'intervalle $[0 ; L-1]$ où $f(x, y) = 0$ correspond au blanc et $f(x, y) = L-1$ au noir.

➢ *Histogram equalization*

Parmi les transformations pertinentes pour nos données, nous pouvons songer à la technique d'égalisation des histogrammes pour augmenter les contrastes : en sortie nous obtenons $g(x, y)$ une image à fort contrastes.
Posons :

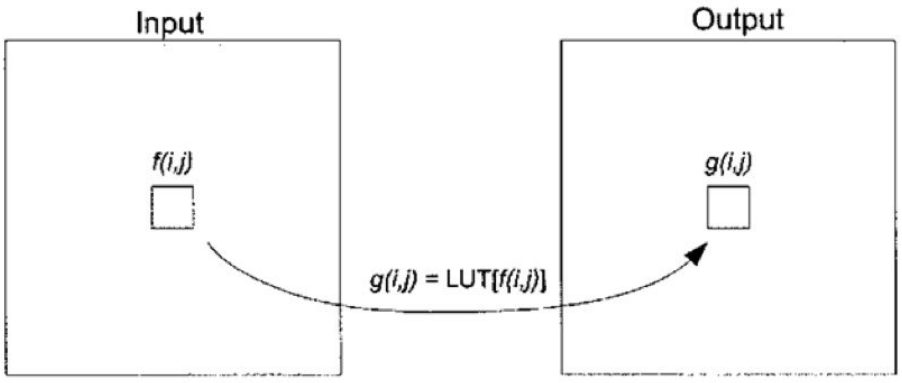
- $r_k = f(x, y)$
- $s_k = g(x, y) = T(r_k)$
- $n_k = h(r_k)$ le nombre de pixels de l'image f prenant la valeur r_k et h la fonction discrète donnant l'histogramme -qui aurait pu être p la fonction normalisée

Une image à fort contraste présente un histogramme à allure uniforme, ainsi on cherche à changer la valeur de l'intensité de f telle que l'image en sortie g ait un tel histogramme (ce qui n'est pas notre le cas pour le moment comme illustré par la **Figure 4**). Pour cela on peut poser :

$$g(x, y) = s_k = (L-1) \sum_{j=0}^{k} p(r_j)$$

➢ *Windowing*

Le concept de base du windowing est d'appliquer une fonction de transformation linéaire gray-scale, sous la forme d'une table de consultation appelée LUT, qui se caractérise par deux paramètres : la window et le level. Au final on obtient une mise en évidence des intensités des pixels correspondant à un sous-ensemble de la gamme dynamique entière, au détriment des pixels qui se situent en dehors de cette gamme. Le schéma suivant illustre la technique de traitement où chaque pixel de l'image en entrée f est mis en correspondance avec le pixel correspondant de l'image en sortie g, et la nouvelle intensité du pixel prend une valeur spécifiée par la LUT :



Classiquement on définit la fonction de transformation linéaire en échelle de gris (grey-scale) par la largeur de la LUT où la pente est non nulle (window) et le centre de ce même segment (level). Ainsi, maintenir la window constante tout en ajustant le level a pour effet de déplacer la partie de pente non nulle de la transformation. Et de la même façon, la modification du paramètre de la window élargit ou réduit la plage d'intensité du pixel.
Le schéma suivant donne 3 exemples de fonction de transformation LUT :

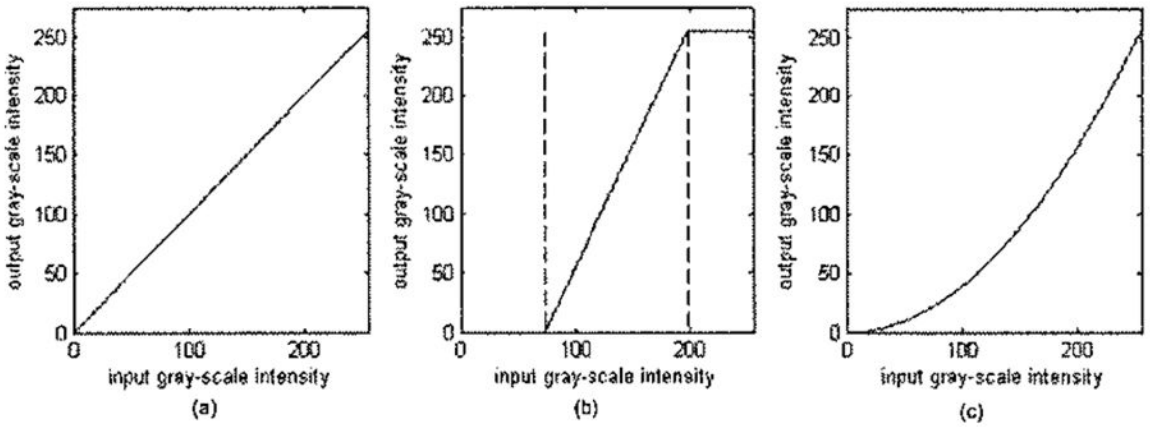


Figure 7 - exemples de fonctions de transformation LUT (a) Fonction d'identité nominale : [0-255] vers [0-255]. (b) Fonction linéaire : vers [74-197] (c) Fonction de transformation gamma non linéaire

Dans notre cas, nos données étant des CT scans, réaliser du window processing doit être envisagé car pour de telles données il est courant d'utiliser le window processing pour mettre en évidence une structure solide au détriment d'une partie molle, ou pour mettre en évidence des anomalies au détriment d'autres structures présentes.

Bibliographie

https://www.info-radiologie.ch/crane-intra-extra-axial.php

https://www.kaggle.com/allunia/rsna-ih-detection-eda/notebook

https://www.kaggle.com/allunia/rsna-ih-detection-baseline

https://www.kaggle.com/felipekitamura/head-ct-hemorrhage-kernel

https://www.kaggle.com/mrdvolk/head-ct-hemorrhage-detection-with-keras

https://fr.wikipedia.org/wiki/Tomodensitométrie

https://www.msdmanuals.com/fr/accueil/troubles-du-cerveau,-de-la-moelle-épinière-et-des-nerfs/accident-vasculaire-cérébral-avc/hémorragie-intracérébrale

https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/
https://projector.datacamp.com/?projector_key=course_7032_ab98136d4a3e96661fdf0930dc5ca587

https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/

https://adventuresinmachinelearning.com/keras-tutorial-cnn-11-lines/

https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610

https://machinelearningmastery.com/how-to-develop-convolutional-neural-networks-for-multi-step-time-series-forecasting/

http://what-when-how.com/embedded-image-processing-on-the-tms320c6000-dsp/windowlevel-image-processing-part-1/

http://www.cse.iitm.ac.in/~vplab/courses/optimization/MAths_IM_DEBLUR_ENH_SD_EDT_2016.pdf