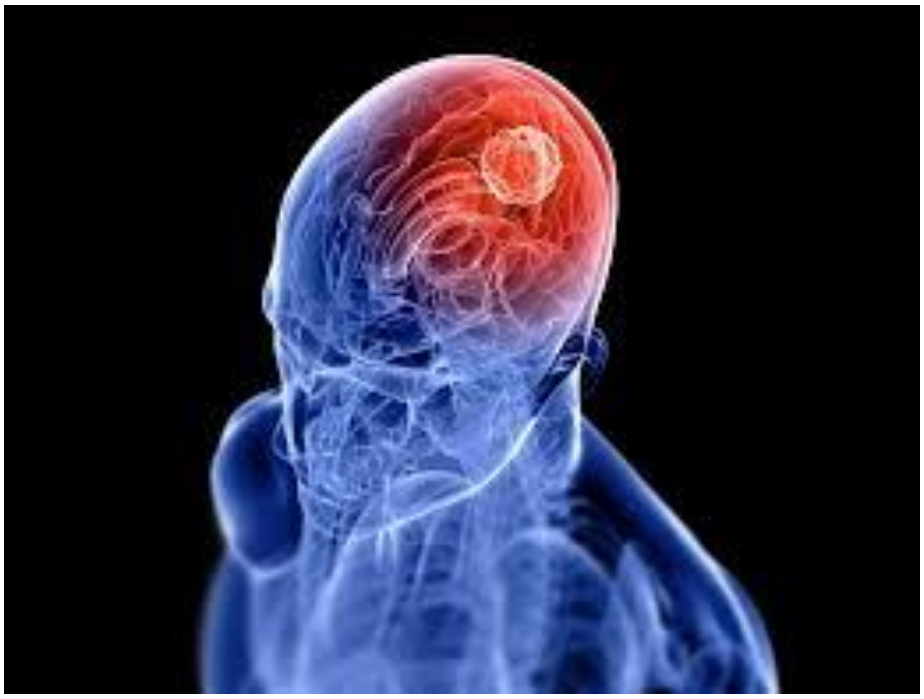


@matherhospital.org

Rapport du projet de Machine Learning

Classification de tumeurs cérébrales par Deep Learning

Louis BERTHIER – Adrien RUGGIERO – Nathan WITKOWICZ

TABLE DES MATIERES

I. Intérêts et motivation pour le projet

1. Nos expériences et ressentis personnels

Nous avons tous les trois suivis le cours d'UE élective « Vision » et nous avons donc un premier intérêt pour le traitement d'images.

Dans le cadre de notre stage de 2ème année, nous nous orientons vers un stage **d'apprentis data scientists** où nous ferons appel au machine learning pour répondre à des problématiques médicales ou liées au marketing. Plus particulièrement, dans le cadre de certaines pathologies, nous allons utiliser du **deep learning** notamment avec des réseaux de neurones convolutifs (CNN). Ainsi, nous avons décidé d'orienter ce projet de machine learning vers un projet plus spécifique en faisant appel à du deep learning pour avoir une **première approche** et **se former en amont** de notre stage.

Nous ne savons pas encore vers quel(s) domaine(s) nous souhaitons nous spécialiser, que ce soit le luxe, la finance, le médical, la robotique, etc. Toutefois nous avons un certain **intérêt pour le domaine de la santé** qui peut, on l'espère, chercher à améliorer le quotidien de personnes souffrants de diverses maladies. En accord avec nos convictions et avec notre futur stage, nous avons décidé de nous orienter vers de la **classification d'images médicales.**

Afin de commencer notre projet, il nous restait donc une seule chose à faire : **choisir la pathologie à étudier**. Personnellement, certains de nos proches ont déjà été affectés par le cancer, succombant ou non à cette terrible maladie. Les datasets étant facilement accessibles, nous avons donc choisi d'étudier plus précisément les **tumeurs liées au cerveau**.

2. Présentation de la pathologie : la tumeur cérébrale

Cependant, qu'est-ce qu'une tumeur cérébrale ? D'après le National Health Service, c'est « une croissance de cellules dans le cerveau qui se multiplient de manière anormale et incontrôlable ». Il existe deux principaux types de tumeurs : les tumeurs cancéreuses (malignes) et les tumeurs non-cancéreuses (bénignes). Les tumeurs cérébrales sont classées selon deux critères à savoir : leur vitesse de croissance et leur probabilité de réapparition après traitement.

De nos jours, il n'existe aucun moyen de prévention face aux tumeurs cérébrales et d'après l'OMS, plus de 300 000 personnes ont été diagnostiquées comme atteintes d'une tumeur cérébrale en 2020 avec plus de **250 000 personnes décédées** de cette maladie la même année. La première idée afin de pallier une telle mortalité est donc d'analyser les images cérébrales pour **détecter la présence ou non d'une tumeur**.

Un tel projet permettrait à des cliniques, à des hôpitaux ou encore à des centres de recherche de classer les différents patients et **de les traiter en conséquence**. Pour les patients, cela **augmenterait leur chance de survie**. En effet, au plus tôt la tumeur est détectée et prise en charge, au plus tôt le patient sera guéri **sans que la tumeur n'ait le temps de devenir cancéreuse**. De plus, cela apporterait une certaine **économie financière et humaine** face aux moyens déployés.

II. Etude et présentation du dataset

1. Constitution du dataset

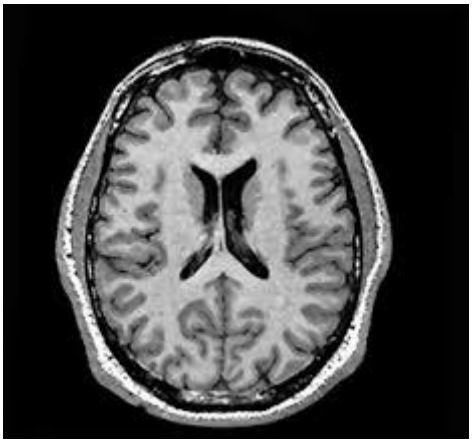
Notre dataset intitulé « Brain MRI Images for Brain Tumor Detection » provient de Kaggle et est disponible à l'adresse suivante :

https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection

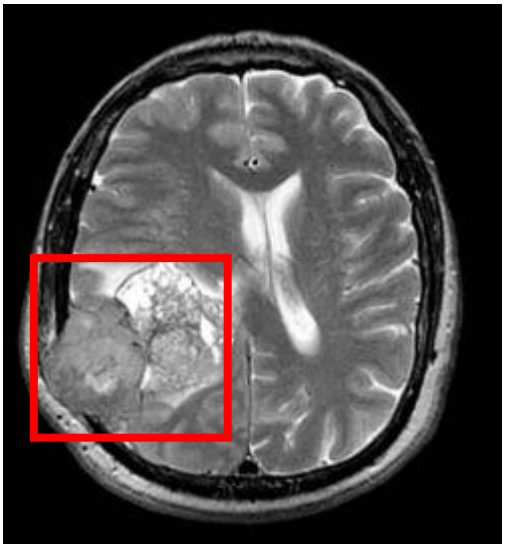
Dedans, nous avons à disposition environ **253 images** principalement au format .jpg avec quelques-unes sous les formats .png et .jpeg. Elles sont toutes en **niveau de gris** (NG). Parmi ces 253 images, on distingue deux catégories :

- **« No »** qui indique l'absence de tumeur
- **« Yes »** qui au contraire indique la présence d'une tumeur

On a 98 images appartenant à la classe « No » et 155 images appartenant à la classe « Yes ». Voici des exemples de nos données :



@25 no.jpg



@Y116.JPG (cadre rouge rajouté manuellement)

2. Remarques concernant le dataset

Comme le montrent ces images, on remarque tout d'abord qu'elles ne sont **pas de même taille**. Également, on remarque que toutes les images n'ont **pas le même niveau d'intensité lumineuse** pour des parties similaires.

Également, dans le cadre d'un **projet scolaire**, le dataset est **adapté** pour développer un modèle et l'entraîner. Cependant, d'un point de vue applicatif pour des **centres de recherches** comme énoncés en partie I, le nombre d'images contenues dans le dataset est clairement **insuffisant**. Il faudrait des milliers voir des millions d'images pour entraîner de manière correcte notre modèle, afin **d'ajuster les hyperparamètres** et affiner et **d'améliorer notre métrique d'évaluation**.

III. Définition de notre problème de Machine Learning

Comme expliqué à travers notre motivation pour le projet, nous allons traiter ce problème à travers du Machine Learning et plus précisément à travers une méthode de plus en plus utilisée de nos jours : le **Deep Learning**. Nous allons pour cela faire appel à des **réseaux de neurones convolutifs** (CNN).

Dans le cadre de notre projet, nous devons donc catégoriser une image **présentant une tumeur** ou non.

- En entrée : 253 images en NG dont les dimensions varient d'une image à l'autre
- En sortie : {présence de tumeur, absence de tumeur}

Le deep learning nous permet de distinguer informatiquement les **caractéristiques à analyser** alors qu'avec du machine learning c'est à l'Homme de les préciser. En effet, il n'y a pas de « feature extraction » en deep learning, donc on ne choisit pas les variables (caractéristiques) avec lesquelles on va travailler : c'est le CNN qui s'en charge.

Également nous travaillons sur des images, ainsi l'architecture CNN nous permet de **conserver des propriétés liées à la spatialité**.

IV. Notre méthodologie et notre protocole pour résoudre ce problème

Maintenant que nous avons clairement défini notre projet et notre problème, nous devons nous intéresser à sa résolution. Lorsque l'on traite un problème de machine learning, la tâche principale après avoir défini les objectifs et le problème est de **préparer notre dataset** afin d'avoir une exploitation efficace et cohérente.

1. Répartition de nos images

Afin de respecter la méthodologie pour accepter ou non un modèle, nous procédons premièrement à un découpage de notre dataset initial que l'on répartie en 3 catégories à savoir :

- Données pour **l'entraînement** du modèle (à hauteur de 70%)
- Données pour **la validation** du modèle (20%)
- Données pour **tester** le modèle (10%)

2. Opérations sur les images

Afin de correctement utiliser nos données, nous avons choisi de les traiter en amont de l'implémentation dans notre modèle. En effet, on remarque que les images sont différentes :

- Les **dimensions** : hauteur et largeur
- Les **couleurs** : on a des images RBG

Pour faire face à cela, nous avons implémenté plusieurs procédures afin de rendre les images comparables :

- On **normalise** chaque image afin de se ramener à **l'échelle [0,1]**. Cela permet **d'être dans les plages de nos fonctions d'activation** et d'**accélérer le processus d'apprentissage**. Le processus de normalisation permet d'**étendre la plage de valeurs** d'une image à l'ensemble des valeurs disponibles.
- On **redimensionne** nos images à une taille unique. Cette unicité permet **l'utilisation des images au sein de notre CNN** et les dimensions choisies (150x150) permettent un **apprentissage plus rapide**. En effet, plus les dimensions sont petites, moins il y a de pixels et plus le réseau de neurones apprend rapidement.
- On **transforme** nos images RBG vers des images **en niveau de gris**. Nos images ont beau être RBG, quand on les visualise, elles ne **présentent pas de « couleurs ».** Également, en transformant nos images en niveau de gris, cela sera **moins coûteux** pour notre CNN étant donné que l'**on passe de 3 plans à un seul**.

3. Modélisation de notre réseau de neurones

Comme défini plus haut, nous avons opté pour **un réseau de neurones convolutifs (CNN)**. Ainsi nous avons la possibilité d'ajouter des couches de neurones successives afin de réaliser **différentes opérations**. Notre CNN prend bien évidemment en entrée une image en NG de taille 150x150.

Dans un premier temps, nous avons opté pour l'utilisation de **5 couches** différentes :

- **Conv2D** : cette couche produit autant de « feature-maps » qu'elle est composée de neurones. Chaque feature-map possède des caractéristiques différentes de l'image.
- **BatchNormalization** : souvent placée entre deux couches, elle nous permet de normaliser les valeurs prises en entrée pour améliorer le traitement de la couche suivante.
- **MaxPooling2D** : cette couche a pour objectif d'extraire des tendances ou des motifs des données. Elle va donc extraire la valeur maximale afin de conserver seulement les informations importantes de chaque feature map en entrée.
- **Flatten** : son objectif est d'aplatir le tenseur en entrée afin de se ramener à un vecteur donc une dimension : 1-D.
- **Dense** : chaque neurone de cette couche reçoit une entrée de tous les neurones de la couche précédente (fully connected). C'est une couche qui permet de prédire l'appartenance à une catégorie à partir des sorties de la couche précédente.

N'ayant aucune connaissance vis-à-vis du nombre de paramètres ou encore du nombre et du types des couches, nous avons discuté de tout ceci avec M. Xu afin de nous éclairer.

4. Early stopping

On implémente un moyen de régulariser notre modèle afin d'**éviter l'overfitting**. Cette nouvelle variable utilise le « early stopping » afin d'arrêter l'entraînement si notre métrique (ici 'val_loss') **cesse de s'améliorer au bout d'un certain nombre d'epochs** (ici 15). Cette variable est essentielle dans les "gros" projets pour **permettre des économies de temps** (en jours ou en mois) **et d'argent** (milliers ou millions d'euros).

5. Entraînement du modèle

Dans le cas où on aurait besoin de conserver le modèle on implémente une **étape de sauvegarde et de réutilisation**. On entraîne notre modèle à l'aide de l'architecture de notre CCN sur le jeu d'entraînement et ensuite on cherche à le valider via le second jeu : celui de la validation en profitant de l'early stopping.

Par la suite, on affiche les courbes des résultats où on expose différentes métriques à la fois pour l'entraînement et pour l'évaluation, ce qui nous permet **d'étudier le comportement de notre modèle** : bon prédicteur ? overfitting ? underfitting ?

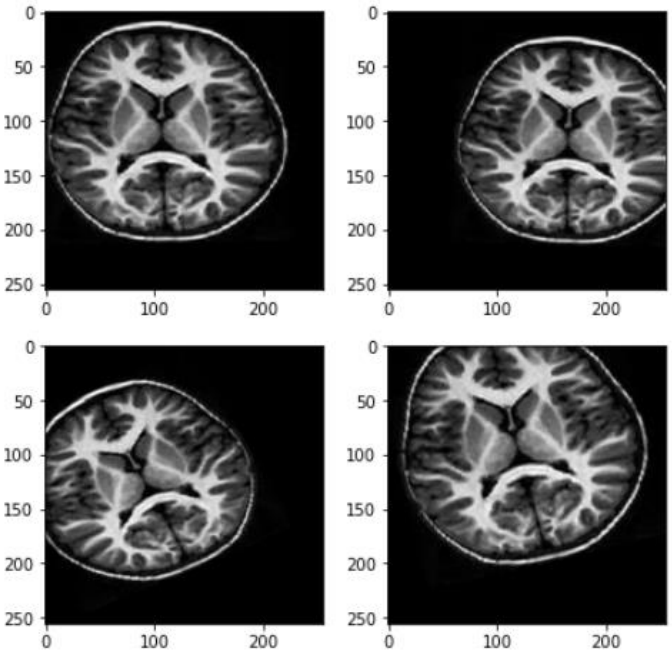
Nos deux principales métriques sont **'loss'** notre fonction coût qu'on définit comme 'binary_crossentropy' et **'AUC'** (Area Under the Curve).

Ensuite, on applique tout simplement notre modèle à notre jeu de données restant, conservé pour le test, et on observe les résultats.

V. Implémentation de différentes méthodes de résolution

1. La Data Augmentation (DA)

Comme on peut le remarquer, notre dataset contient peu de données, nous avons donc cherché à l'augmenter. Une procédure simple consiste à utiliser la **data augmentation** (DA) qui vient **modifier artificiellement** certaines images de notre dataset pour l'entraînement afin **d'augmenter le nombre de données à entraîner**. En plus de l'augmentation, cela nous permet de diversifier les données sans réellement en collecter. Finalement, l'objectif est de **limiter l'overfitting** puisque notre CNN considérera les images générées comme **distinctes**.



@Exemples issus de notre code où l'on applique différentes transformations à une même image

Ici, c'est un simple exemple. En effet, nous n'appliquons pas toutes ces transformations aussi marquées à notre training set.

Dans notre cas nous avons utilisé les transformations suivantes :

- Des **rotations** de l'image
- Des **retournements verticaux**
- Des **retournements horizontaux**
- Un **zoom** sur l'image
- Un **rognage** de l'image

Nous avons essayé de **limiter les effets** de ces transformations pour ne pas trop modifier l'image. En effet, une combinaison d'un zoom, d'un rognage et d'une rotation peut nous faire **perdre notre point/zone d'intérêt** au sein de l'image à savoir la tumeur notamment en découpant sur la partie du cerveau où elle se trouve. On peut donc se retrouver avec une image considérée comme sans tumeur alors qu'elle est labélisée comme en possédant une : on ajoute donc des **faux négatifs**.

2. Répétition et comparaison de modèles

Nous souhaitions comparer certains modèles, nous nous sommes donc focalisés sur **4 études** à savoir un modèle comportant l'optimisateur **'Adam'** et un modèle comportant l'optimisateur **'SGD'** avec pour les 2 un cas **sans DA** et un cas **avec DA**. Sur deux ordinateurs nous avons donc cherché à comparer ces deux modèles : l'un conservant 'Adam' et l'autre utilisant 'SGD'.

Afin de rendre les résultats **plus pertinents**, nous avons décidé de **répéter ces expériences une dizaine de fois**. Cela nous permettra d'avoir une idée de **quelle configuration** représente le **meilleur modèle**. Pour s'assurer de comparer le même dataset avec les deux optimisateurs, nous fixons un **aléa différent pour chaque expérience mais identique sur les 2 ordinateurs**.

3. Matrice de confusion et évaluation

Pour comparer l'ensemble des modèles étudiés à travers la dizaine d'expérience, nous faisons appel à **la matrice de confusion** pour récupérer certaines métriques supplémentaires. Plus précisément on s'intéresse **au rappel**. Dans notre modèle on considère les images portant une tumeur comme notre **élément pertinent**. Le rappel va donc nous indiquer **si l'ensemble des personnes qui ont besoin d'être traitées sont identifiées** alors que la précision va nous indiquer **si l'on traite des personnes qui n'ont pas besoin de l'être**.

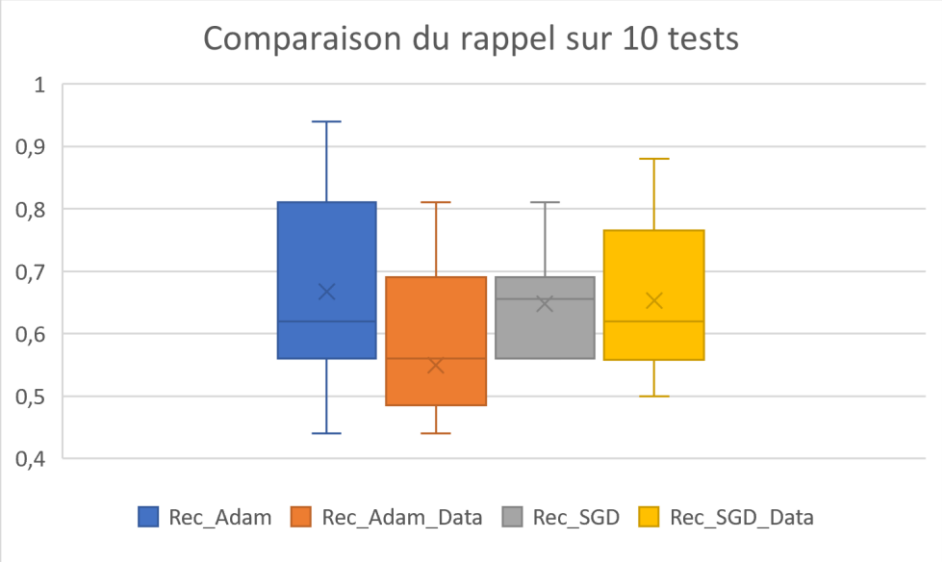
On cherche à maximiser nos chances de **traiter toutes les personnes qui portent la pathologie,** d'où l'intérêt pour le rappel.

Également, nous nous sommes intéressés à la métrique 'AUC' plutôt que 'acc' car elle est **plus pertinente** dans notre étude, notamment à travers l'étude des **faux négatifs**. En effet, elle prend en compte **les probabilités de prédiction** pour chaque classe, et est **plus représentative et robuste pour des datasets de petite taille**.

VI. Discussion de nos résultats

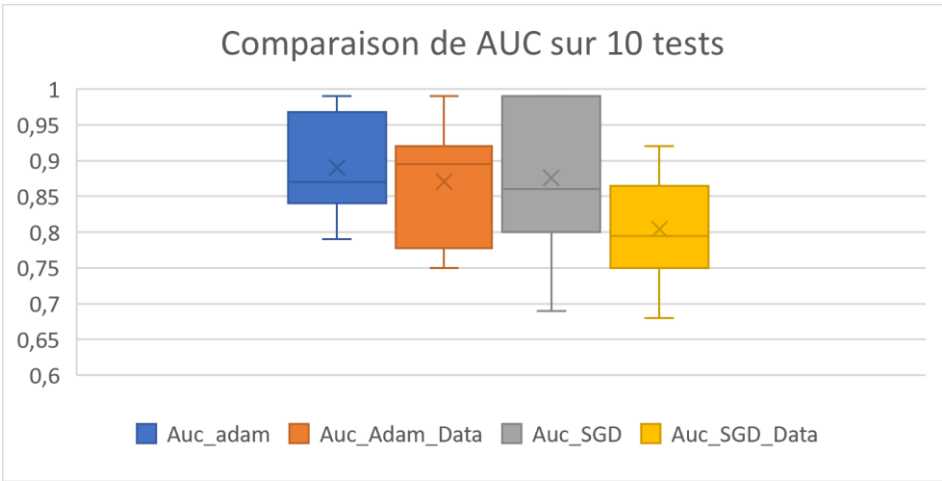
1. Les résultats obtenus

Voici un graphique issu de nos tableaux Excel comparant nos différents modèles à travers des boîtes à moustache sur la métrique à laquelle nous accordons le plus d'importance, le rappel *(cf Annexe 1.)* :



Pour le rappel, on distingue que le meilleur modèle correspond à **l'optimisateur 'SGD' sans la data augmentation** notamment grâce à sa médiane et à la taille de la boîte (resserrée).

Ce modèle **maximise** comme voulu **nos chances de traiter les personnes ayant une tumeur**.



Pour AUC, la boîte du SGD n'est clairement **pas un résultat favorable**, notamment en le comparant à Adam.

On a une boîte beaucoup **trop étendue ce qui n'assure pas la stabilité des prédictions**. Cependant, on préfère **accorder de l'importance au rappel plutôt qu'à AUC**.

2. Les limites de notre modèle

A travers notre projet, nous avons pu constater différentes limites à savoir :

- Premièrement, nous avons **un manque de données** en entrée. Seulement 253 images pour entraîner un modèle de Deep learning, ce n'est **pas suffisant**.
- **Quels paramètres devons-nous modifier afin d'améliorer notre architecture et notre modèle ?** Devons-nous modifier le nombre de couches et leur type ? Devons-nous changer le nombre de neurones ? Devons-nous choisir un taux d'apprentissage différent ? Devons-nous garder notre fonction coût ?

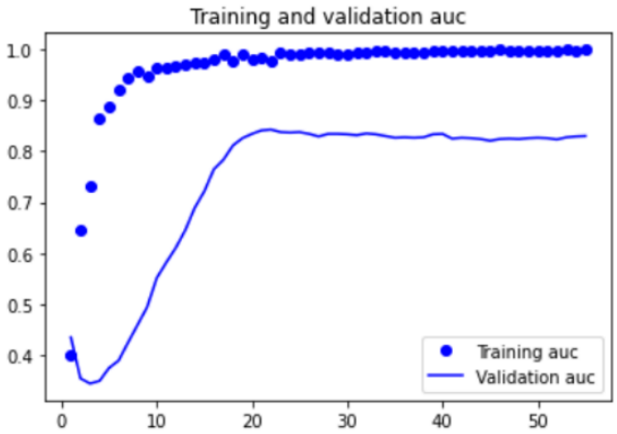
VII. Conclusion sur le projet

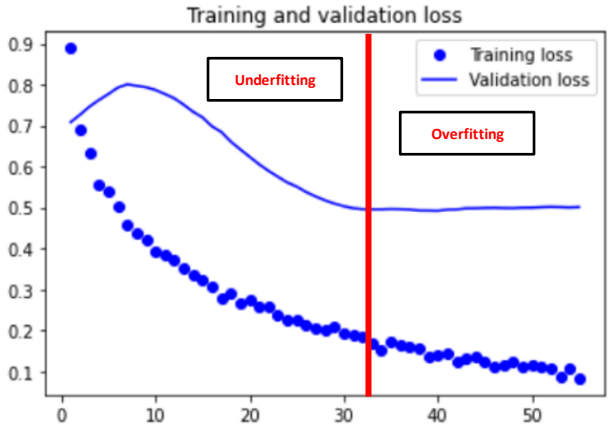
1. La solution retenue

À travers notre problématique et les résultats proposées précédemment, nous décidons de rester sur une architecture composée des points suivants :

- Optimisateur : **SGD**
- Data Augmentation : **Sans**

Voici un exemple des résultats et des courbes que nous obtenons avec un tel choix :





On voit que notre modèle est en underfitting avant une 30aine d'epochs et qu'il commence à être en overfitting une fois qu'on dépasse les 33 – 35 epochs. Comme prévu, notre early-stopping arrête l'entraînement aux alentours des 50 epochs, soit 15 epochs plus tard ce qui correspond à la patience indiquée dans notre algorithme.

2. Nos observations et préconisations

On a pu remarquer que la data augmentation **n'a pas été un grand succès**. En général, la data augmentation est un processus utilisé dans le **domaine de l'imagerie médicale** après discussion avec un data scientist chez Sanofi. Nous ne comprenons donc pas pourquoi dans notre cas, ce procédé se révèle peu probant. **L'explication peut-elle provenir de notre dataset original ?**

Afin de mieux discuter les résultats, il est aussi intéressant de regarder sur Kaggle **les différents codes et modèles** proposées par d'autres utilisateurs. On peut notamment se rendre compte que c'est **l'apprentissage par transfert** qui offre les meilleurs résultats :

	Model Name	Precision (with Max AUC)	Recall (with Max AUC)	Max AUC	Accuracy	Mean Precision	Mean Recall	Mean AUC	Mean Accuracy
0	Simple CNN	0.818182	0.870968	0.896435	0.80	0.773411	0.854839	0.802037	0.753
1	MobileNetV2	0.966667	0.935484	0.988115	0.94	0.956002	0.908065	0.969567	0.916
2	InceptionResNetV2	0.964286	0.870968	0.964346	0.90	0.893168	0.877419	0.931919	0.854
3	VGG16	0.961538	0.806452	0.974533	0.86	0.896765	0.879032	0.937818	0.854

VIII. Annexes

1. Notre fiche excel où nous avons annoté l'ensemble de nos expériences et des métriques associées :

ADAM				
Yes (1)	f1	Auc_adam	Acc	Rec_Adam
1	0,47	0,79	0,41	0,44
2	0,57	0,87	0,44	0,62
3	0,56	0,87	0,48	0,56
4	0,76	0,84	0,7	0,81
5	0,58	0,99	0.52	0,56
6	0.56	0,85	0.48	0,56
7	0.76	0,9	0,7	0,81
8	0,73	0,99	0,66	0,75
9	0,61	0,96	0,52	0,62
10	0,71	0,84	0,56	0,94
Moyenne	0,62375	0,89	0,56	0,67
Médiane	0,595	0,87	0,54	0,62

SGD				
Yes (1)	f1	Auc_SGD	Acc	Rec_SGD
1	0,65	0,82	0,407	0,69
2	0,6	0,99	0,519	0,61
3	0,76	0,77	0,703	0,81
4	0,59	0,69	0,44	0,69
5	0,6	0,98	0,56	0,56
6	0,69	0,88	0,63	0,69
7	0,62	0,81	0,56	0,62
8	0,56	0,99	0,48	0,56
9	0,53	0,99	0,41	0,56
10	0,63	0,84	0,52	0,69
Moyenne	0,62	0,88	0,52	0,65
Médiane	0,61	0,86	0,5195	0,655

seed = n° Test

ADAM (data augmented)				
Yes (1)	f1	Auc	Acc	Rec_Adam_Data
1	0,59	0,77	0,48	0,62
2	0,47	0,85	0,41	0,44
3	0,58	0,91	0,52	0,56
4	0,55	0,78	0,51	0,5
5	0.22	0,91	0.48	0,12
6	0.58	0,91	0.52	0,56
7	0.76	0,75	0.70	0,81
8	0,65	0,99	0,55	0,69
9	0,69	0,95	0,63	0,69
10	0,52	0,88	0,45	0,5
Moyenne	0,58	0,87	0,51	0,55
Médiane	0,58	0,895	0,51	0,56

SGD (Data Augmented)				
Yes (1)	f1	Auc	Acc	Rec_SGD_Data
1	0,53	0,81	0,41	0,56
2	0,65	0,83	0,558	0,68
3	0,61	0,68	0,52	0,62
4	0,72	0,75	0,63	0,81
5	0,55	0,88	0,55	0,5
6	0,69	0,78	0,59	0,75
7	0,58	0,86	0,52	0,56
8	0,55	0,92	0,44	0,55
9	0,59	0,78	0,48	0,62
10	0,78	0,75	0,7	0,88
Moyenne	0,63	0,80	0,54	0,65
Médiane	0,6	0,795	0,535	0,62