Compétition Kaggle - Decembre 2024 : Identification de maladies de la rétine par images OCT

Thierry POEY - N°20307205 et Noé POULAIN - N°20304939

1. Introduction

Le projet donné est une classification d'images médicales, où l'objectif est de concevoir un algorithme capable de classifier avec précision des images de tomographie par cohérence optique (OCT). Ces images représentent des coupes transversales de la rétine et permettent de diagnostiquer les cas suivantes : néovascularisation choroïdienne, œdème maculaire diabétique, drusen, ou rétine saine. Pour les besoins de ce rapport, les pathologies mentionnées seront respectivement remplacées par (CN, DME, Drusen et Healthy)

Les données fournies sont des ensembles d'images (28x28 pixels sur 1 canal gris) avec leurs étiquettes associées pour l'entraînement. Une approche méthodique a été adoptée, en combinant des modèles pré-entraînés sur ImageNet-1k et ImageNet-21k puis en les appliquant le fine tuning. Diverses techniques d'optimisation et de régularisation, telles que l'optimisation bayésienne, la data augmentation et l'utilisation de learning rate schedulers, ont permis d'améliorer les performances.

Parmi les modèles testés, le Vision transformers (ViT), EfficientNet-B0, et un modèle hybride combinant ResNet-50 et EfficientNet-B0 se sont distingués par leurs performances. À noter cependant que bien que le modèle hybride a montré un plus fort potentiel, son optimisation reste perfectible, un point qui sera abordé ultérieurement.

Les résultats ont été évalués à l'aide de métriques telles que l'accuracy et le macro F1-score, menant à une accuracy finale sur le meilleur modèle de 0.81488 sur l'ensemble privé de la compétition.

2. Conception des fonctionnalités

Durant cette phase, Grad-CAM a été utilisée. Grad-CAM permet de visualiser les régions les plus influentes pour les prédictions du modèle. Cette approche nous a aidé à identifier les erreurs de classification et à valider la pertinence des transformations appliquées sur les données.

2.1. Prétraitement des modèles

Les modèles pré-entraînés sont initialement entraînés sur des datasets comme ImageNet1K et ImageNet21K. On utilise les poids déjà optimisés pour détecter des caractéristiques comme les contours, textures ou motifs complexes. Par ailleurs, les couches d'entrée de ces modèles appliquent une standardisation des canaux RGB selon des moyennes et des écarts-types fixes, pour obtenir une homogénéité des données en entrée. Cette étape est essentielle pour maximiser l'efficacité des phases d'entraînement et de fine-tuning lors du transfer learning que nous verrons dans la prochaine partie.

2.2. Augmentation des données

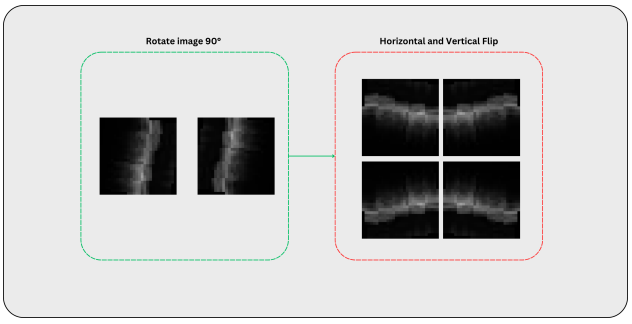


Figure 1. Exemples d'augmentation des données

La data augmentation a deux objectifs principaux : diversifier le jeu de données en ajoutant de nouveaux exemples et rendre le modèle plus robuste face au risque d'overfitting. Dans cette étude, 5 méthodes ont été utilisées :

Homogénéité : Comme vu dans la section précédente, les poids des modèles ont été pré-entraîné sur des datasets normalisé. Pour garantir une conformité des données, il est essentiel d'appliquer les mêmes standardisations. Cela incluait la conversion en RGB ainsi que la normalisation des donnée.

Recadrage : Les informations importantes pour détecter les différentes maladies peuvent être présentes à différents endroits dans l'image. On sélectionne aléatoirement une sous-région de l'image d'origine, simulant un zoom sur une

partie de celle-ci. Cette sous-région est ensuite redimensionnée pour correspondre à la taille d'entrée attendue par les modèles. Ce procédé permet au modèle de concentrer son analyse sur une zone locale spécifique plutôt que sur l'ensemble de l'image.

Rotation : Cette technique élargit efficacement la perspective du modèle, lui donnant l'adaptabilité nécessaire pour reconnaître les objets indépendamment de leur orientation.

Flip : Applique un effet miroir sur l'image originale à la fois horizontalement et verticalement.

Modification de la couleur: Créer des petites variétés de lumières et de contraste pour simuler des conditions d'éclairage variées.

En raison de la faible résolution des images (28x28 pixels), de petites modifications introduites par ces transformations ont eu un impact disproportionné sur les résultats de nos expériences. Leurs utilisations ont été contrôlée pour limiter ces effets négatifs tout en maximisant leurs bénéfices.

2.3. Masquage de zones non pertinentes

Pour une analyse précise des maladies rétiniennes, il est essentiel de conserver les informations provenant des couches inférieures de la rétine, telles que l'ELM (Membrane Limitante Externe) et la zone ellipsoïde des photorécepteurs. La méthode consistait à masquer les zones non pertinentes, représentant entre 10 et 30% de la hauteur totale des images. Ce processus permet de réduire l'impact du bruit tout en maintenant les données essentielles à une détection efficace des pathologies. Toutefois, en raison de la grande variabilité des données, cette approche a parfois conduit à la suppression d'informations essentielles pour la détection des pathologies. Nous avons préféré ne pas la conserver pour nos modèles.

2.4. Équilibrage des classes

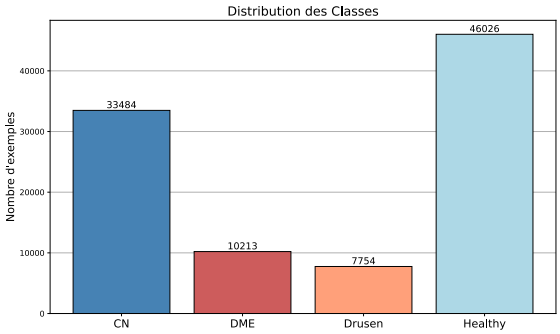


Figure 2. Impact du déséquilibre des classes sur les performances du modèle.

Afin de traiter le déséquilibre des classes 2 et 3, nous avons expérimenté deux approches complémentaires d'équilibrage des classes :

SMOTE : Cette méthode crée artificiellement de nouvelles instances pour les classes minoritaires en interpolant les données existantes avec leurs k-plus proches voisins.

Undersampling : Cette méthode réduit la taille des classes majoritaires en supprimant aléatoirement des exemples, jusqu'à obtenir un équilibre avec la classe minoritaire.

Malgré l'application de ces deux techniques, les résultats obtenus n'ont pas montré d'amélioration significative. Dans les prochaines sections, nous analyserons en détail les réels défis qui ont pu limiter l'efficacité de ces approches et proposerons des solutions adaptées.

3. Algorithmes

Nous nous concentrerons ici uniquement sur les algorithmes ayant démontré des performances significatives, ainsi que sur le modèle de baseline. Les modèles qui n'ont pas été optimisés ou qui ont affiché des performances insuffisantes, tels que MobileNetV3, ResNet-18, DenseNet121 et VGG16, seront exclus de cette discussion.

3.1. Multi-Layer Perceptron (baseline)

Dans cette partie, le MLP a été implémenté à partir de zéro en utilisant uniquement numpy.

Le modèle est composé d'une architecture avec :

- **Une couche d'entrée** de 784 neurones (28x28 pixels des images d'entrée). Les pixels sont aplatis en vecteurs 1D pour correspondre au format attendu d'un MLP.
- **Une seule couche cachée** de 64 neurones pour capturer des caractéristiques non linéaires.
- **Une couche de sortie** comprenant 4 neurones, chacun correspondant à une classe de pathologie, avec une activation softmax.

Les poids des couches ont été initialisés en utilisant la méthode de Xavier, pour garantir que les activations restent dans des plages raisonnables au départ.

L'entropie croisée a été choisie pour mesurer la différence entre les prédictions probabilistes du modèle et les vraies classes :

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \cdot \log(\hat{y}_{i,c} + \epsilon)$$

avec ϵ, pour éviter un $log(0)$.

La fonction softmax utilisée dans ce modèle transforme les scores bruts en probabilités normalisées.

- **Soustraction de** $\max(x)$ **:** Cette étape empêche les exponentielles de faire un overflow.
- **Normalisation :** Les scores sont divisés par leur somme, pour une sortie comprise entre 0 et 1.

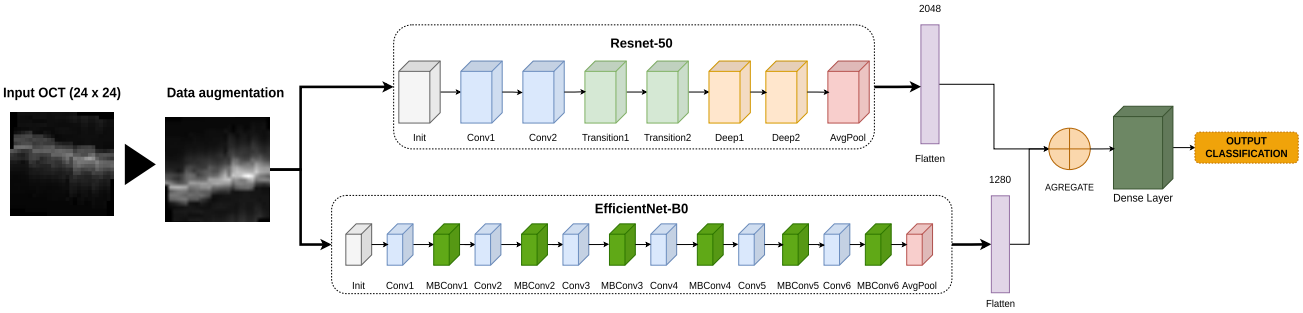
Pour l'entraînement, la descente de gradient stochastique (SGD) a été employée, avec un taux de dropout fixé à 0,2 pour réduire le sur-apprentissage et améliorer la généralisation.

3.2. Vision Transformer (ViT)

Le Vision Transformer (ViT) a été utilisé comme modèle avancé pour la classification des images rétiniennes, donnant l'opportunité d'explorer le fonctionnement des transformers. Contrairement aux architectures convolutionnelles (CNN), le ViT segmente les images en patches, les traite comme des séquences, et utilise un mécanisme d'attention pour analyser les relations globales entre les patches. Tout ceci a été simplifié grâce à la librairie *transformers*.

L'entraînement a été réalisé avec l'optimiseur AdamW, adapté aux transformers, et un scheduler cosine pour ajuster le taux d'apprentissage.

3.3. Hybride (EfficientNet-B0 + ResNet-50)



Le modèle hybride (figure 3.3) combine les architectures EfficientNet-B0 et ResNet-50 pour tirer parti de leurs forces complémentaires. L'objectif est d'optimiser l'extraction des caractéristiques des images en utilisant les avantages spécifiques de chacune.

EfficientNet-B0 est particulièrement performant pour capter les informations globales des images grâce à sa structure légère et efficace, adaptée aux ressources limitées. En parallèle, ResNet-50 performe dans l'extraction des détails locaux grâce à ses connexions résiduelles (d'où le nom donné).

Les caractéristiques extraites par ces deux modèles sont ensuite concaténées pour former un vecteur unique de grande dimension (3328), qui est traité par une série de couches : une première couche linéaire réduit cette dimension à 512, suivie d'une activation ReLU pour capturer des relations non linéaires. Puis une couche de dropout, avec un taux de 30 %, est utilisée pour réduire le risque d'overfitting. Enfin, une dernière couche linéaire qui produit les prédictions finales sur les quatre classes.

Ce modèle se distingue par sa capacité à fusionner des informations globales et locales, offrant ainsi une vision plus complète et précise des images.

L'entraînement du modèle repose sur l'optimiseur Adam, avec un taux d'apprentissage initial fixé à 10^{-4}, et un scheduler cosine qui ajuste dynamiquement ce taux. La régularisation est effectué par le dropout pour limiter l'overfitting.

Sans prétention, le modèle présenté s'est avéré être le meilleur que nous ayons conçu. Malheureusement, sa découverte est intervenue trop tardivement, ce qui n'a pas permis d'effectuer une optimisation complète. Je reste néanmoins convaincu que ce modèle pourrais passer un pallier dans cette compétition, une affirmation que je détaillerai ultérieurement.

4. Méthodologie

Nous aborderons ici les choix adoptés pour l'entraînement du modèle. Notamment la répartition des données, les stratégies de régularisation, les optimiseurs, les hyperparamètres sélectionnés et l'optimisation bayésienne.

4.1. Stratified Sampling

Pour notre étude, nous disposions d'un ensemble d'environ 97 000 images. Cependant, un tel volume de données est trop important pour optimiser efficacement nos modèle. Pour pallier ce problème, nous avons adopté une

stratégie d'échantillonnage stratifié. Cette méthode garantit une répartition équilibrée des classes dans l'ensemble d'entraînement, réduisant ainsi les biais de sélection et assurant que chaque classe contribue proportionnellement à la construction du modèle. Afin de limiter ces coûts, nous avons restreint nos expérimentations à 20% du jeu de données global. Cette approche nous permet de tester rapidement différents ensembles d'hyper-paramètres et d'obtenir une première estimation de la convergence du modèle, avant de passer à un entraînement complet sur l'ensemble des données disponibles.

4.2. Validation Croisée Stratifiée

Une validation croisée en k-folds stratifiée a été adoptée pour garantir une évaluation robuste des performances. En divisant les données en cinq sous-ensembles équilibrés, cette méthode a permis une estimation plus fiable de la généralisation du modèle.

4.3. Utilisation de Grad-CAM

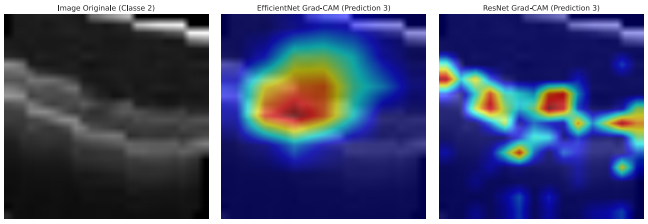


Figure 3. CamGrad Misclassified

Dans notre étude, Grad-CAM a été utilisé pour visualiser les régions d'intérêt sur lesquelles le modèle s'appuie pour effectuer ses prédictions. L'implémentation a ciblé les dernières couches convolutionnelles des architectures utilisées pour EfficientNet-B0 et ResNet-50. Ces couches ont été choisies car elles fournissent les activations finales des convolutions.

Lors des prédictions, les images mal classifiées ont été identifiées et des visualisations Grad-CAM ont été générées pour ces cas spécifiques. Les images sont normalisées et affichées en parallèle, montrant les zones activées par les deux architectures. Cette analyse a permis de repérer les scénarios où le modèle se concentrait sur des zones non pertinente au lieu des régions rétiniennes essentielles.

Cette approche a directement influencé notre processus d'entraînement. Par exemple, les ajustements apportés aux transformations des données ont été guidés par ces visualisations. De plus, les résultats Grad-CAM ont confirmé la pertinence des couches pré-entraînées utilisées. Ils ont également mis en évidence les avantages du modèle hybride en combinant les forces du ResNet-50 et EfficientNet-B0 et aussi les faiblesses d'un EfficientNet-B0 seul.

4.4. Optimisation Bayésienne

Bien que la plupart de nos modèles sont pré-entraîné et que le fine-tuning a déjà été testé sur de nombreux articles. Il est important d'effectuer son propre fine tuning sur son propre dataset avec les plages de valeurs recommandés. L'optimisation bayésienne, réalisée avec *optuna* nous a aidé à choisir *learning rate* et le *weight decay*. Cette méthode s'est avérée particulièrement bénéfique pour les modèles comme EfficientNet où des interactions complexes entre hyper-paramètres influencent fortement les performances. Il s'avère que pour EfficientNet, le learning rate se rapproche de 1e-4 et le weight decay de 0.

4.5. Gestion du taux d'apprentissage

Deux stratégies principales ont été testées pour ajuster dynamiquement le taux d'apprentissage pendant l'entraînement :

- **Cosine Annealing** : Réduction progressive du taux d'apprentissage selon une fonction cosinus, facilitant une convergence plus douce et réduisant les risques d'oscillations tardives.
- **StepLR** : Diminution du taux d'apprentissage à des intervalles définis.

4.6. Régularisation des poids

La régularisation via le paramètre *weight decay* introduit une contrainte qui pénalise les poids élevés, favorisant des solutions plus généralisables. Une plage entre 10^{-5} et 10^{-2} a été explorée pour adapter la régularisation selon le modèle utilisé. Le modèle hybride et le ViT ont tendance à overfit. Il était donc utile d'explorer les plages de valeurs sans trop freiner l'apprentissage.

4.7. Gel et Dégel des couches

Les couches pré-entraînées ont été initialement gelées pour préserver les caractéristiques déjà apprises sur ImageNet. Après 10 % de l'entraînement, les couches ont été dégelées. Cette stratégie a notamment permis au modèle un overfitting prématuré.

4.8. Précision mixte (AMP)

L'utilisation de l'AMP a énormément réduit les besoins en mémoire GPU et accéléré les calculs. En combinant des représentations FP16 pour les opérations moins sensibles et FP32 pour celles nécessitant une grande précision, cette technique a optimisé les ressources. En utilisant les GPU de VastAI, j'ai constaté que cela fonctionne pour de nombreux GPU.

4.9. Early Stopping

Un mécanisme d'early stopping a été mis en place pour arrêter l'entraînement si les performances sur l'ensemble de validation ne s'amélioraient plus après 5 à 10 époques. Ce processus a permis de limiter le temps de calcul inutile.

4.10. Équilibrage des classes

Pour gérer les déséquilibres entre les classes, une pondération adaptée (*weighted loss*) a été appliquée en fonction de la distribution des classes.

4.11. Optimiseurs

Plusieurs optimiseurs ont été testés pour ajuster les poids du modèle. La plupart sont juste des recommandations trouvées sur internet :

- **AdamW et Adam**: Une convergence rapide et sa gestion efficace des régularisations.
- **RMSprop**: Particulièrement adapté pour les données non stationnaires, mais nécessitant des ajustements fins pour éviter des oscillations instables.
- **SGD avec Momentum**: Bien que plus lent à converger, il peut être meilleurs que les autres optimiseurs mais il reste difficile à optimiser.

L'optimiseur Adam a été retenu pour la plupart des expérimentations en raison de la rapidité de sa convergence.

5. Résultats

Les résultats ont été obtenus en utilisant différentes méthodes de prétraitement, des configurations d'hyperparamètres et des algorithmes de classification. L'analyse inclut une comparaison entre le Multi-layer Perceptron, le ViT, EfficientNet-B0 et le modèle hybride. Sera présenté en premier les performance mesurée par le F1-score et l'accuracy sur les modèles finaux.

Note : Les performance sont mesurée sur 20 epochs avec seulement 20% du dataset (voir sections précédentes).

5.1. Comparaison des Modèles

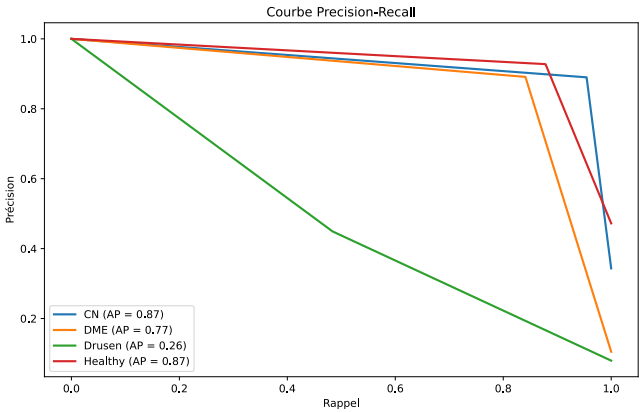
Résultats finaux :

Table 1. Performance Finale des modèles de classification

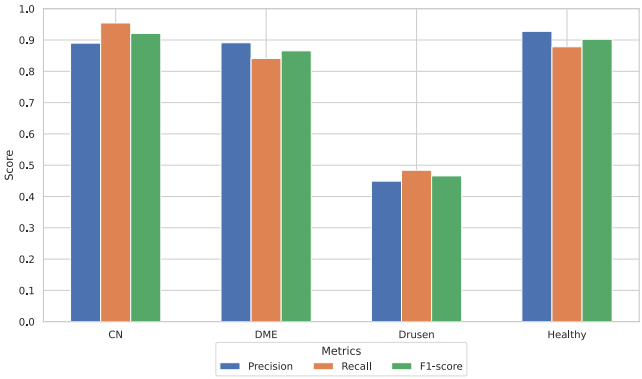
Modèle	F1-score (Validation)	Accuracy (Validation)	Accuracy (Kaggle)
MLP		0.6867	0.541214
ViT	0.8019	0.8766	0.81488
EfficientNet-B0	0.80	0.8641	0.79275
Hybride	0.8033	0.8756	0.76257

L'analyse des résultats obtenus pour les différents modèles de classification (ViT, EfficientNet-B0, Hybride) en fonction des méthodes de prétraitement appliquées révèle plusieurs tendances intéressantes. Nous analyserons ces résultats dans les prochaines sous-sections.

ViT



La courbe de Précision-Rappel (Figure 5.1) illustre la capacité du modèle à maintenir un équilibre entre précision (précision positive) et sensibilité (taux de vrais positifs). Le comportement du ViT dans cette courbe met en évidence sa capacité à minimiser les faux négatifs pour les classes minoritaires tout en maintenant une haute précision pour les classes majoritaires. Cette performance est très positive pour les maladies rares comme *DME* et *Drusen*, qui sont souvent confondues avec les autres classes dans des modèles moins performants.



Grâce à son processus d'attention, il capte les relations entre les différentes régions d'une image, ce qui est particulièrement pertinent pour détecter des maladies rétiniennes où des signes subtils peuvent apparaître dans différentes zones de l'image.

De plus, l'analyse des métriques montre une faible variance entre les classes, indiquant que le modèle ne privilégie pas les classes majoritaires aux dépens des minoritaires.

Cependant, quelques limitations subsistent. Par exemple, des erreurs de classification entre *Drusen* et *DME* peuvent indiquer que certaines caractéristiques discriminantes sont trop similaires ou mal représentées dans les données d'entraînement. Cela suggère que des ajustements supplémentaires, comme une augmentation des données sur les classes minoritaires, pourraient encore affiner les performances du modèle.

Hybride

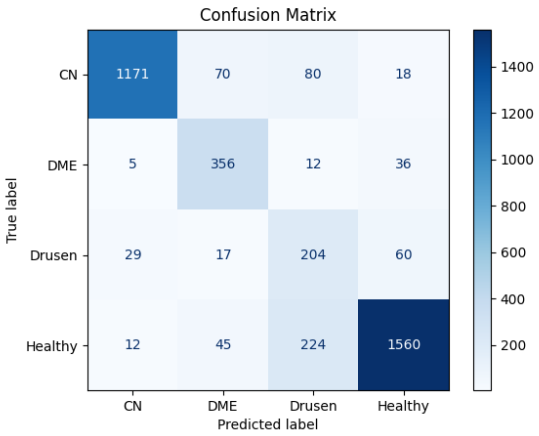


Figure 4. Matrice de confusion pour le modèle hybride.

La matrice de confusion (Figure 4) révèle que le modèle hybride parvient à classifier efficacement la plupart des classes, bien que des confusions subsistent, notamment entre les classes *Drusen* et *DME*.

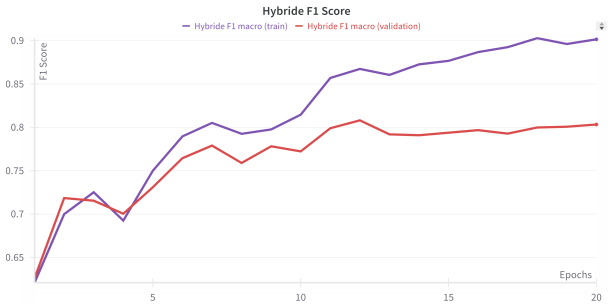


Figure 5. Source : Wandb

L'évolution du score F1 macro (Figure 5) montre une convergence stable sur l'ensemble d'entraînement, avec une légère divergence sur l'ensemble de validation après 15 époques. Cette tendance indique que le modèle overfit sur l'ensemble d'entraînement.

6. Discussion et Perspectives

Notre premier modèle, basé sur ViT (Vision Transformer), a été une véritable surprise. Facile à implémenter et déjà optimisé, il répondait à de nombreuses attentes. Cependant, son optimisation restait limitée, et il s'est avéré extrêmement gourmand en données. Pour explorer d'autres solutions, nous avons utilisé la bibliothèque *timm*, ce qui nous a permis de tester plusieurs modèles. Parmi eux, l'EfficientNet-B0 s'est distingué comme une option prometteuse. Nous avons commencé à l'optimiser, mais l'ajout des Grad-CAM a rapidement révélé une limitation majeure : même après des ajustements poussés, l'architecture peinait à exploiter les informations car le contenu des images reste limité. Néanmoins, cette étape a marqué une avancée significative, car elle nous a permis d'implémenter et de tester une variété de méthodologies.

Un point critique a été l'identification de la classe "Drusen" comme étant particulièrement problématique. Cette classe était souvent confondue avec la classe "Healthy", ce qui influençait fortement les performances du modèle.

Vers la fin de la compétition, nous avons exploré une approche hybride combinant les atouts de l'EfficientNet-B0 et du ResNet-50. Ce modèle hybride a considérablement amélioré les performances globales, notamment grâce à une synergie efficace entre les deux architectures. Pour contextualiser, le meilleur score observé sur le leaderboard Kaggle atteignait environ 0,85. Ce résultat, bien qu'ambitieux, semblait cohérent avec nos observations : bien que la classe "DME" (minoritaires) soit facilement identifiable, la classe "Drusen" restait beaucoup plus difficile à classifier avec précision.

Je suis convaincu que ce modèle hybride, grâce à sa capacité à capturer des relations complexes, est capable d'exceller sur l'ensemble de validation comme il a pu le montrer sur l'ensemble d'entraînement, à condition d'ajuster correctement les hyper-paramètres.

Pour exploiter pleinement le potentiel de cette architecture hybride, il serait judicieux d'explorer davantage l'impact des stratégies de fusion et de réduction de la dimensionnalité. Par exemple, l'intégration de techniques comme la PCA ou des auto-encodeurs pourrait permettre de réduire le bruit dans les caractéristiques.

References

https://huggingface.co/blog/fine-tune-vit

https://arxiv.org/abs/2411.00873

https://arxiv.org/abs/1905.11946

https://link.springer.com/article/10.1007/s00521-024-09820-w