Université Caen Normandie

Projet Piotox
Master Informatique
Rapport

Rapport Projet Annuel

Etudiants :
Kaoutar Arbouch
Antoine Mezeray

Encadrants :
Alexis Lechervy
Youssef Chahir
Juliette Thariat

31 janvier 2023

Résumé

Le présent rapport résume le travail effectué dans le cadre du projet annuel réalisé durant le premier semestre de la dernière année d'études du Master Image et Données Multimédia (IDM) de l'Université Caen Normandie.

L'objectif principal du sujet consiste à trouver une solution pour segmenter des images brutes médicales obtenues avec la méthode Optical Coherence Tomography angiography (OCTa), afin de quantifier les nerfs optiques fins et gros.

Pour atteindre notre objectif, nous avons travaillé sur deux approches : les méthodes OCTA-Net et VAFF-Net, qui sont des algorithmes d'apprentissage profond. Ce rapport commence par un état de l'art, suivi des réalisations que nous avons faites. Il se terminera par des résultats et une conclusion.

Mots clés : Segmentation des images médicales, images OCTa, apprentissage profond, OCTA-Net, VAFF-Net.

Abstract

This report resumes the work produced during the first semester of the second year of the "Image et Données Multimédia" master of Caen Normandie's University.

The main goal of this project is to find a solution to segment raw medical images obtained thanks to the Optical Coherence Tomography angiography (OCTa) technique, in order to quantify thin and thick optical vessels.

To achieve our goal, we worked on two different approaches : OCTA-Net and VAFF-Net, which are both deep learning algorithms. This report will present the state of the art and our realizations. Finally, we will present the results we've obtained and we will conclude.

Keywords : Segmentation of medical images, OCTa images, Deep Learning, OCTA-Net, VAFF-Net.

Liste des abréviations

Abréviation	**Signification**
AUC	Area Under the Receiver Operating Characteristic (ROC) Curve
CNN	Convolutional Neural Networks
DVC	Deep Vascular Complexes
FAZ	Foveal Avascular Zone
FN	False negative
FP	False positive
GPU	Graphics Processing Units
IOU	Intersection over Union
IVC	Inner Vascular Complexes
OCTa	Optical Coherence Tomography angiography
ROSE	Retinal OCTa Segmentation
SVC	Superficial Vascular Complexes
TN	True Negative
TP	True Positive
VAFF	Voting based Adaptive Feature Fusion multi-task network

TABLE 1 – Tableau d'abréviations.

Table des matières

Table des figures

Liste des tableaux

1 Introduction générale

Dans le cadre de notre projet annuel, nous devions choisir un projet parmi plusieurs proposés. Nous avons choisi de travailler sur le projet "PIOtox". Ce projet a été créé grâce à la collaboration entre le laboratoire GREYC et le centre François Baclesse de Caen.

Ce projet nous a tout particulièrement intéressé, d'un côté parce que sa problématique touche au monde médical et il a un objectif concret à atteindre, et de l'autre côté, ce projet nécessite de faire du traitement d'images.

1.1 Objectif du projet

Notre objectif principal est de trouver une solution permettant de faire la segmentation d'images obtenus avec la méthode Optical Coherence Tomography angiography (OCTa) automatiquement grâce au machine learning. Les images OCTa capturent les nerfs optiques d'un œil. Si nous arrivons à trouver et mettre en place une solution efficace pour résoudre ce problème, alors nous pourrions utiliser ces images segmentées afin d'établir une corrélation entre ces images et le champ visuel d'un patient.

1.2 Outils de développement

Nous avons principalement travaillé sur deux algorithmes codés en Python. Nous avons été amenés à modifier et améliorer les codes de ces algorithmes, et nous avons pu travailler avec les bibliothèques python suivantes :

- Pytorch ;
- Pytorch lightning.

De plus, nous avons utilisé les outils suivants pour mener à bien notre projet :

- la plateforme Google Colab sur Internet (notebooks Python en ligne avec un accès GPU) ;
- l'application Jupyter Notebook,
- la plateforme GitHub, pour rendre les améliorations apportées sur le code facile à partager et conserver un historique des modifications ;
- la plateforme Weights & Biases (WandB), afin de pouvoir visualiser la progression des algorithmes au cours du temps (loss, image en sortie, ...) et rendre ces données facilement accessible pour les encadrants.
- l'application PyCharm pour modifier le code en Python ;
- l'application Anaconda Navigator, afin de créer des environnements spécifiques pour chaque algorithme et pouvant gérer l'accès à d'autres applications comme Jupyter Notebook et PyCharm.

2 État de l'art

2.1 Méthodes de segmentation historiques

2.1.1 Le modèle CNN : classification d'images [1]

Pour comprendre les modèles de segmentation d'images en apprentissage automatique, nous devons nous pencher d'abord sur une technique de classification d'images, et tout particulièrement le Convolutional Neural Networks (CNN). Le modèle présenté dans la partie suivante (U-Net) se base sur l'architecture du CNN.

Pour résoudre le problème de classification des images, les multilayer perceptrons (MLP) ne sont pas les plus efficaces : tous les neurones de ce modèle sont complètement connectés les uns aux autres, donc chacun d'entre eux va traiter le même nombre d'entrées. Or, les images représentent une quantité de données très importantes : une image en couleur de taille 32 x 32 pixels représente 32*32*3 = 3072 entrées, et une image en couleur de taille 256 x 256 représente 196 608 entrées pour chacun de ces neurones.

C'est pour cela qu'un autre type de modèle a été créé : le modèle CNN. Il s'inspire du principe de reconnaissance d'information visuelle des animaux. Les images naturelles ont une logique de construction dans l'espace (deux pixels adjacents sont fortement corrélés), et des opérations comme la convolution peuvent prendre avantage de cette caractéristique.

Un CNN est construit grâce à plusieurs couches : principalement de convolution, de max pooling (sous-échantillonage, réduit la taille de l'entrée), et aussi de correction (ReLU), des couches fully connected (FC), et de loss.

La convolution permet de créer une nouvelle image à partir d'une image d'entrée : pour chaque pixel de l'image d'entrée, on calcule une nouvelle valeur à partir de ce pixel de ses pixels voisins en appliquant un coefficient à chacun de ces pixels. Plusieurs paramètres de convolution sont possibles : par exemple, prendre uniquement un pixel sur deux de l'image d'origine, prendre des pixels voisins dans une plus grande zone, ou ajouter des pixels au bord de l'image de base pour agrandir la dimension de l'image obtenue après convolution.

Le max pooling permet de réduire la dimension de la matrice en entrée et de ne garder que les valeurs les plus fortes dans chaque zone de la matrice. Par exemple, une matrice de taille 4x4 contient 16 valeurs : le max pooling découpe cette matrice en 4 sous-matrice de taille 2x2, et ne garde que la valeur maximum de chacune de ces sous-matrice pour obtenir une matrice de taille 2x2.

Cependant, nous nous intéressons dans notre cas à la segmentation et non à la classification d'images. Mais une des solutions les plus connues pour la segmentation d'images est basée sur le modèle CNN : le modèle U-Net.

2.1.2 Le modèle U-Net : segmentation d'images [2]

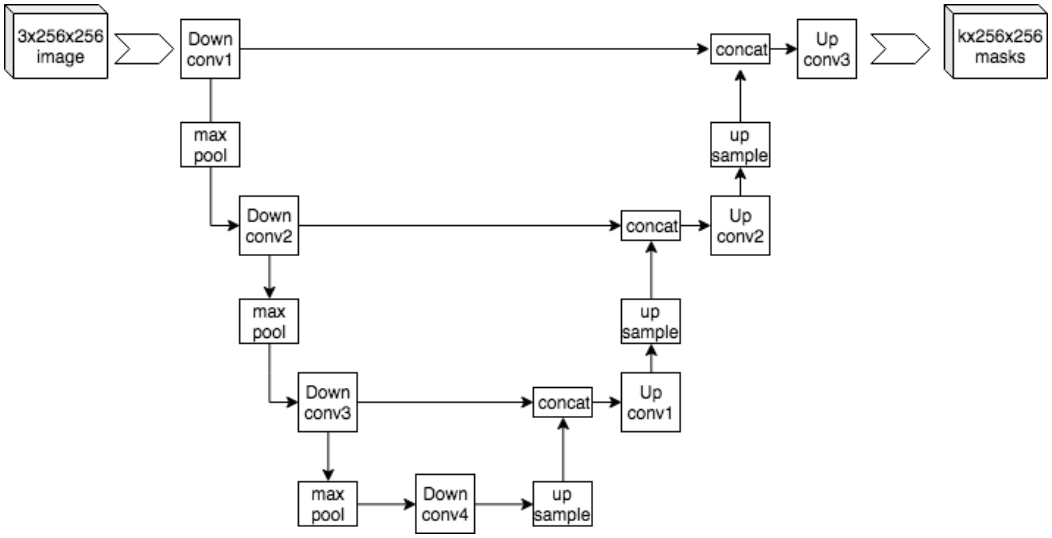

FIGURE 1 – Exemple d'un modèle U-Net

Le modèle U-Net a été développé afin de segmenter des images biomédicales, et sa première partie est basé sur le même principe qu'un CNN. Le modèle U-Net est symétrique : la première partie est un encodeur (on passe d'une image en entrée à un vecteur de caractéristique), et la deuxième partie est un décodeur (on passe de ce vecteur de caractéristique obtenu vers une nouvelle image). La première partie utilise des couches de convolutions et de max pooling ; la deuxième partie va donc utiliser des couches de déconvolution et de suréchantillonage pour effectuer les opérations inverses. De plus, des connections directes entre les encodeurs et décodeurs (avec des sorties et entrées de même taille) sont mises en place pour permettre à ce modèle d'être plus performant.

2.2 Méthodes de segmentation pour notre problématique

Ceci nous amène à notre problématique. Nous avons effectué un état de l'art des solutions déjà existantes au tout début du projet. Nous avions retenus les articles suivants :

Date de publication	Dataset	Code	Objectif	Avantages et Inconvénients
31/08/22	ROSE	✓	Segmentation automatique d'images OCTa (vaisseaux), FAZ, et jonctions des vaisseaux Méthode « VAFF-Net » (pytorch) [3]	+ : Présente des résultats avec des méthodes déjà existantes, en général a de meilleurs résultats que les autres méthodes
07/12/20	ROSE	✓	Segmentation automatique d'images OCTa sur les DVC et SVC Méthode « OCTA-Net » (pytorch) [4]	+ : Se compare aux autres méthodes existantes
04/05/20	OCTA-500	✓	Segmentation automatique d'images OCTa et FAZ Méthode « IPN » (tensorflow) [5]	- : Pas d'accès libre à l'article scientifique - : Utilise tensorflow v1, une bibliothèque de deep learning python qui n'est plus d'actualité
14/12/20	OCTA-500	✓	Segmentation automatique d'images OCTa et FAZ Méthode « IPNV2 » (pytorch) [6]	+ : Se compare aux autres méthodes existantes
22/07/22	ROSE		Génération d'images synthétiques d'OCTa [7]	+ : Peut être utilisées en tant que dataset pour entraîner des algorithmes
18/10/21			Comparer les méthodes existantes traitant de la segmentation et classification automatique d'images OCTa [8]	+ : Liste assez exhaustive - : Date « déjà » d'il y a un an, à comparer avec des méthodes plus récentes ? - : Les résultats des méthodes ne sont pas forcément comparables (ex : certains ne vont avoir que l'accuracy quand d'autre n'auront que Dice)

TABLE 2 – Méthodes de segmentation pour notre problématique

Parmi tous ces articles, nous avons retenus quatre solutions : "OCTA-Net", "VAFF-Net", "IPN" et "IPN-V2". Ces solutions ont toutes l'avantage d'être publiques, nous pouvons récupérer les algorithmes sur leur github dédié. Toutes ces solutions sont des algorithmes d'apprentissage profond (deep learning). Les deux premières solutions sont basés sur la base de données ROSE, tandis que la seconde est basée sur la base de données OCTA-500. Ces bases de données sont accessibles après avoir effectué une demande auprès de leur auteur.

2.2.1 Base de données ROSE

- ROSE-1 [9] : 117 images OCTa de 39 patients (dont 26 malades), séparés en 90 images d'entraînement et 27 images de test. Angiogrammes en face des Superficial, Deep and Inner Vascular Complexes (SVC, DVC et IVC).
  - système de capture : RTVue XR Avanti SD-OCT system (Optovue, USA) ;
  - résolution 304 x 304 pixels ;
  - zone capturée de 3x3 mm^2, centrée sur la fovéa ;
  - vérité terrain : annotations manuelles par des cliniciens et des "experts en image".
- ROSE-2 [9] : 112 images OCTa de 112 yeux , séparés en 90 images d'entraînement et 22 images de test.
  - système de capture : logiciel Spectralis (Heidelberg Engineering, Heidelberg, Germany) ;
  - résolution 840 x 840 pixels (après avoir été agrandi) ;
  - zone capturée de 3x3 mm^2, centrée sur la fovéa ;
  - vérité terrain : annotations manuelles par un ophtalmologiste expérimenté à l'aide d'un logiciel interne codé avec Matlab.
- ROSE-O [10] : 117 images OCTa des RV, RVJ et FAZ.
  - système de capture : Optovue Avanti RTVue XR avec le logiciel AngioVue (Optovue, Fremont, USA) ;
  - résolution 304 x 304 pixels ;
  - vérité terrain : annotations manuelles.

2.2.2 Base de données OCTA-500

- 500 ensemble d'images (volumes OCT et OCTa) répartis entre deux sous-ensemble [11] :
  - OCTA 6M avec 300 ensemble d'images ;
  - FOV : 6 x 6 x 2 mm ;
  - Volume : 400 x 400 x 640 pixels.
  - OCTA 3M avec 200 ensemble d'images ;
  - FOV : 3 x 3 x 2 mm ;
  - Volume : 304 x 304 x 640 pixels.
- Labels sous forme de texte (par exemple, âge ou sexe) ;
- Labels sous forme d'image (segmentation des vaisseaux et des FAZ).

Au final, nous avons choisi les solutions basés sur la base de données ROSE, car cette base de données est proche de la base de données du centre François Baclesse. De plus,

nous avions essayé de tester l'alogrithme IPN, mais il n'était plus pertinent car il utilisait une bibliothèque de deep learning trop ancienne (tensorflow v1).

2.3 Détail des algorithmes retenus

2.3.1 OCTA-Net

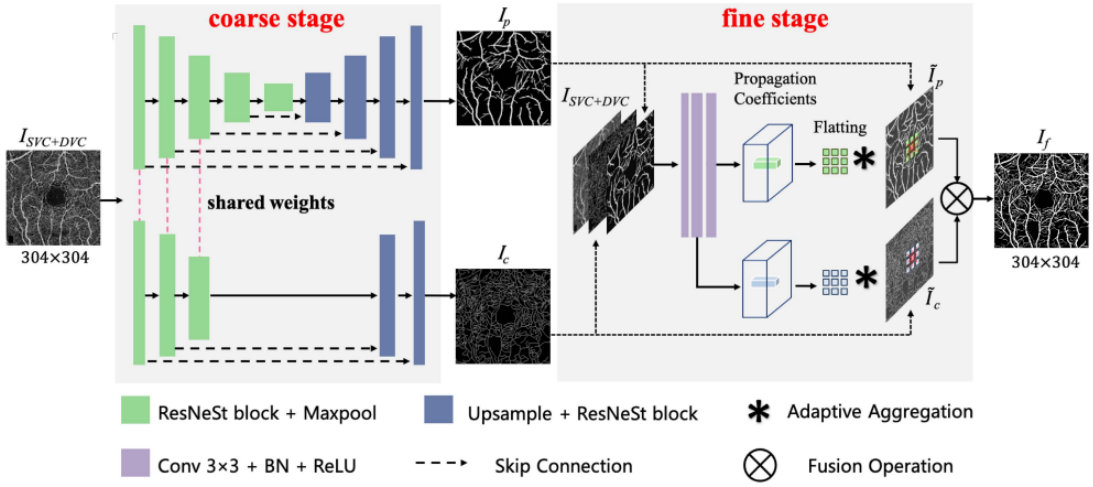

FIGURE 2 – Architecture du réseau OCTA-Net (avec un exemple de l'angioram SVC+DVC dans le jeu de données ROSE-1). [3]

L'algorithme OCTA-Net se compose de deux parties :

1. **Coarse stage** : a pour objectif de générer deux cartes de confiance préliminaires qui fait la segmentation des vaisseaux au niveau du pixel (I_p) et la Segmentation des vaisseaux au niveau de la ligne centrale (I_c) [3]. C'est la partie qui sert à séparer les vaisseaux en deux images, une pour les vaisseaux mince et l'autre pour les vaisseaux épais.

   Coarse stage utilise "l'erreur quadratique moyenne" (MSE) comme fonction de perte :

   $$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(p_i - g_i)^2$$

   avec N le nombre de tous les pixels, pi et gi représentent respectivement le i-ème pixel de l'image de prédiction et la vérité terrain.

2. Et le modèle **Fine stage** qui est ensuite utilisé pour fusionner ces informations et obtenir la segmentation finale affinée des images OCTA (I_f).
   Le modèle prend en entrée la concaténation des vaisseaux prédites au niveau des pixel et au niveau de la ligne centrale et aussi l'image OCTA d'origine(de canal unique), il améliore la précision des cartes de confiance générées par le coarse stage,

en utilisant des coefficients adaptatifs. Il utilise un mini réseau comprenant trois couches de convolution (de noyaux 3×3), et des couches BN et ReLU sont adoptées après chaque couche de convolution. Finalement, les cartes au niveau du pixel (I_p) et au niveau de la ligne centrale (I_p) affinées sont ensuite fusionnées en une image complète de segmentation des vaisseaux, en choisissant la valeur la plus grande des deux cartes à chaque pixel [3].

Pour le fine stage, la fonction de perte MSE a été remplacé par perte de coefficient du Dice pour améliorer la segmentation des vaisseaux lors de la phase finale :

$$L_{Dice} = 1 - \frac{2\sum_{i=1}^{N} p_i g_i + \epsilon}{\sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} g_i^2 + \epsilon}$$

où le paramètre ϵ est une petite constante positive utilisée pour accélérer la convergence d'apprentissage. N le nombre de tous les pixels, pi et gi représentent respectivement le i-ème pixel de l'image de prédiction et la vérité terrain.

2.3.2 VAFF-Net

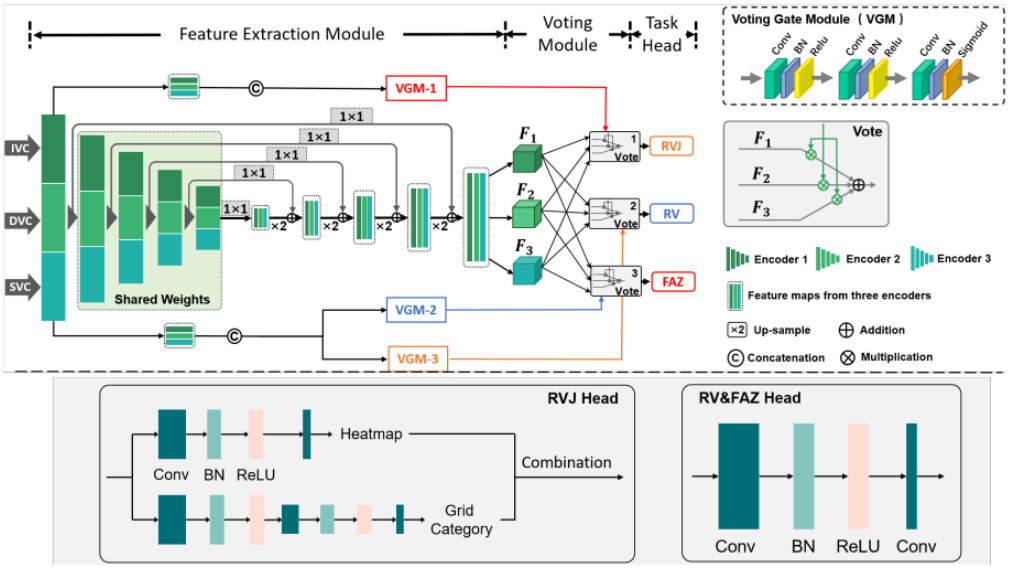

FIGURE 3 – Modèle de VAFF-Net [4]

L'algorithme VAFF-Net est un algorithme multitâches : il permet de faire les segmentations sur les vaisseaux et de la FAZ, et la prédiction des jonctions des vaisseaux en même temps (le but est de repérer les endroits où les vaisseaux se chevauchent où se bifurquent). Il est découpé en 3 partie principale : le module d'extraction de caractéristiques (Feature Extraction Module), le module de vote, et le "Task Head".

Le module d'extraction de caractéristiques est composé de trois sous-parties, chacune étant un extracteur de caractéristique ayant des entrées différentes : images de SVC, DVC ou IVC. Ces trois parties sont des ResNet-50, avec de légères modifications (la première couche de convolution 7 x 7 est remplacé par une couche de convulution 3 x 3).

Le module de vote est aussi découpé en trois sous parties : VGM-1, VGM-1, et VGM-3 (Voting Gate Module). Chaque VGM prend en entrée la concaténation des trois sorties de la première couche de chaque extracteur de caractéristique, et en sortie une porte permettant de sélectionner uniquement les caractéristiques intéressante pour chaque tâche (sur les vaisseaux, la FAZ ou les jonctions). Les caractéristiques extraites sont différentes selon les images d'entrée (par exemple, la segmentation de la FAZ se concentre sur la zone dépourvue de vaisseaux). Les sorties de ces VGM sont multipliées à chaque sortie (carte de caractéristiques) de chaque extracteur de caractéristique, afin d'obtenir une carte de caractéristiques adaptées pour chaque tâche (segmentation des vaisseaux ou de la FAZ, ou la prédiction des jonctions des vaisseaux). Enfin, des dernières couches ("Task Head") traitent les caractéristiques en entrée pour obtenir les images finales générées par l'algorithme.

Le Task Head des des jonctions est un peu plus complexe que celui des segmentations. Le Task Head des jonctions est séparé en deux tâches : la première pour la régression sur une carte de chaleur afin de prédire la position des jonctions, et la deuxième pour la classification par grille (découpage 8 x 8 pixels) permet de déterminer si la jonction est une bifurcation ou un chevauchement de vaisseaux.

Pour le calcul de la loss de la segmentation des vaisseaux et de la FAZ, l'entropie croisée binaire (cross-entropie) est utilisée. Pour celui des jonctions, deux loss sont utilisées : une pour la carte de chaleur et l'autre pour la classification dans une grille. La loss pour la carte de chaleur est une loss des moindres carrées (mean sqaured error), tandis que celle de la carte de chaleur est la suivante :

$$L(Y_{grid}, \hat{Y}_{grid}) = \lambda_A \sum_{i=1}^{S^2} z_i^A (C_i - \hat{C}_i)^2 + \lambda_B \sum_{i=1}^{S^2} z_i^B (C_i - \hat{C}_i)^2 + \sum_{i=1}^{S^2} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

- z_i^A indique si la jonction i apparaît dans la grille i, z_i^B l'inverse ;
- C_i et $\hat{C}_i$ respectivement vérité terrain et score de prédiction de la grille i ;
- $p_i(c)$ et $\hat{p}_i(c)$ respectivement vérité terrain et la probabilité de prédiction pour la classe c pour la grille i ;
- λ_A et λ_B des hyperparamètres, avec des valeurs empiriques de 5 et 1 respectivement.

3 Réalisations

3.1 Algorithme OCTA-Net

Vous pouvez trouver le code originel sur `https://github.com/iMED-Lab/ROSE` et la version de ce code que nous avons amélioré sur `https://github.com/Piotoxproject/octa-net`.

Le data set ROSE-2 a été divisé en deux, 90 images pour l'entraînement et 22 images pour le test. Dans un premier temps nous avons lancé le code original tel quel sur environ 100 epochs sur la plate-forme Google Colab (le code original présentait plusieurs erreurs que nous avons corrigé), et nous avons obtenu des résultats insatisfaisants. Les encadrants nous ont proposé de modifier les paramètres du code et visualiser la performance du modèle (loss, accuracy ...) sur WandB au lieu de Visdom (la méthode de visualisation des résultats présente par défaut dans le code). Ensuite, nous avons fait tourner le code sur 440 epochs et 600 epochs avec des paramètres différents comme nous allons le montrer dans la section 4.1, et nous avons obtenu des résultats assez acceptables que celui de la première.

3.2 Algorithme VAFF-Net

Vous pouvez trouver le code originel sur `https://github.com/iMED-Lab/VAFF-Net` et la version de ce code que nous avons amélioré sur `https://github.com/Piotoxproject/vaff-net`.

Notre première étape a été d'essayer de lancer le code originel tel quel sur Google Colab pour s'assurer de son bon fonctionnement. Ce code présentait déjà une solution pour visualiser les performances et la progression de l'apprentissage au fil des epochs, mais cette méthode ne nous satisfaisait pas : elle utilisait un serveur local, ce qui est assez compliqué à utiliser avec une plateforme en ligne comme Google Colab. Nous avons donc préféré modifier le code pour enlever cette méthode et en implémenter une nouvelle, avec la solution proposée par WandB. Cette solution a été relativement simple à implémenter et nous a permis de contrôler les données envoyées sur le serveur WandB (les images sur la base de train plus les métriques).

La première fois que nous avons lancé le code (sur environ 220 epochs), nous n'avons pas obtenus de résultats satisfaisants. Pour mieux comprendre d'où ce problème provenait, il nous a été proposé de modifier le code pour l'implémenter en "pytorch lightning". Cette méthode permet de rendre le code plus clair et plus efficace : une fois implémentée, nous avons aussi constaté que les epochs étaient un peu plus courts. Réécrire une partie du code en pytorch ligthning nous a permis aussi de pouvoir ajouter une méthode permettant de stopper et reprendre l'apprentissage d'un modèle si besoin (car entraîner un modèle sur 1000 epochs en une fois prend plusieurs heures, dans le cas de cet algorithme).

La plus grosse modification a été de changer le batch size et de le passer de 4 à 1. Cela a été fait surtout pour faire tourner l'algorithme sur une machine personnelle avec une carte graphique récente (8 Go de mémoire) car il était plus facile de travailler en local plutôt qu'avec Google Colab (du à la limitation du temps d'utilisation de GPU par jour, et le besoin de se connecter à Google Drive pour accéder aux fichiers et de réinstaller certaines bibliothèques à chaque fois).

4 Résultats et analyse

Pour les résultats, nous allons principalement utiliser les métriques suivantes :
— La loss, indicateur du bon fonctionnement d'un algorithme d'apprentissage ;
— L'accuracy :

$$\frac{TP + TN}{TP + TN + FP + FN}$$

— Le score de Dice :

$$\frac{2 * TP}{2 * TP + FP + FN}$$

— G-mean score = $\sqrt{Sensitivity * Specificity}$ où $Sensitivity = TP/(TP + FN)$ et $Specificity = TN/(TN + FP)$.
— La balanced accuracy (BACC) :

$$\frac{Sensitivity + Specificity}{2}$$

— AUC : signifie "Surface sous la courbe ROC", qui mesure l'efficacité d'un modèle de classification binaire en comparant la vérité positive et la vérité négative. Il varie entre 0,5 pour un modèle aléatoire à 1 pour un modèle parfaitement précis.

4.1 Algorithme OCTA-Net

Nous avons effectué des tests sur la base de données ROSE-2, la même que celle utilisée dans le travail original. Les résultats obtenus dans le travail original sur ROSE-2 sont :
— Un score de Dice de 70,77%.
— Un score de G-mean de 83,15%.
— Une accuracy de 93,86%.
— Un AUC de 83,03%.

4.1.1 Résultats du premier modèle

Les résultats obtenus du modèle OCTA-net entraîné sur 440 epochs (220 epochs par stage) avec les paramètres :
— learning rate = 1e-4.
— batch size = 2.
sont les suivants :

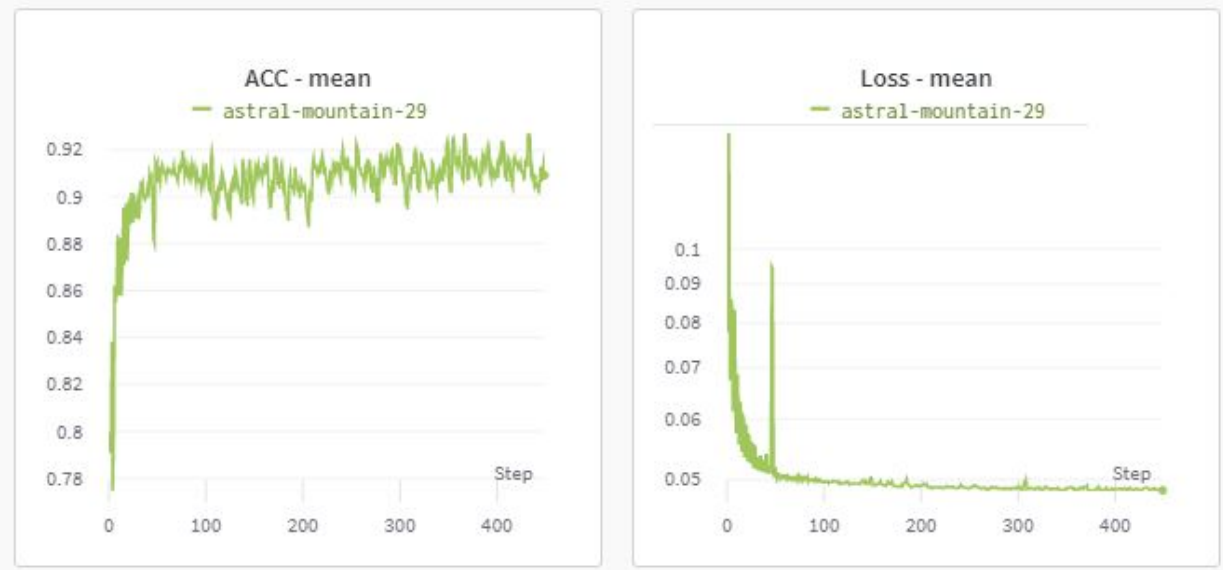

FIGURE 4 – L'accuracy (0.9092) et le loss (0.0484) obtenus après 440 epochs

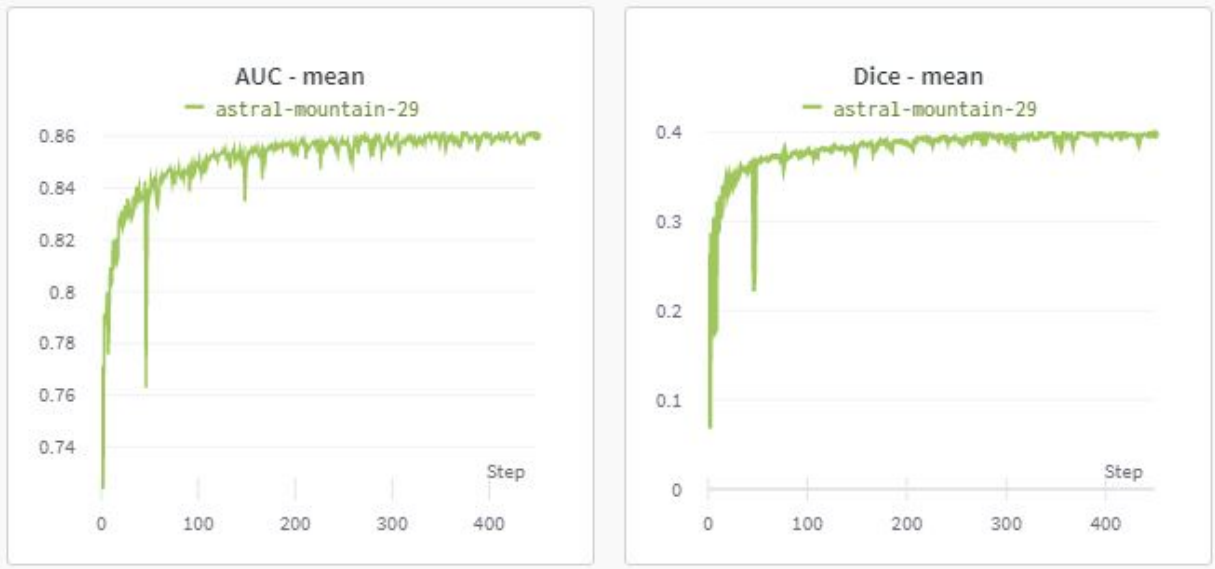

FIGURE 5 – L'AUC (0.8601) le Dice (0.3973) obtenus après 440 epochs

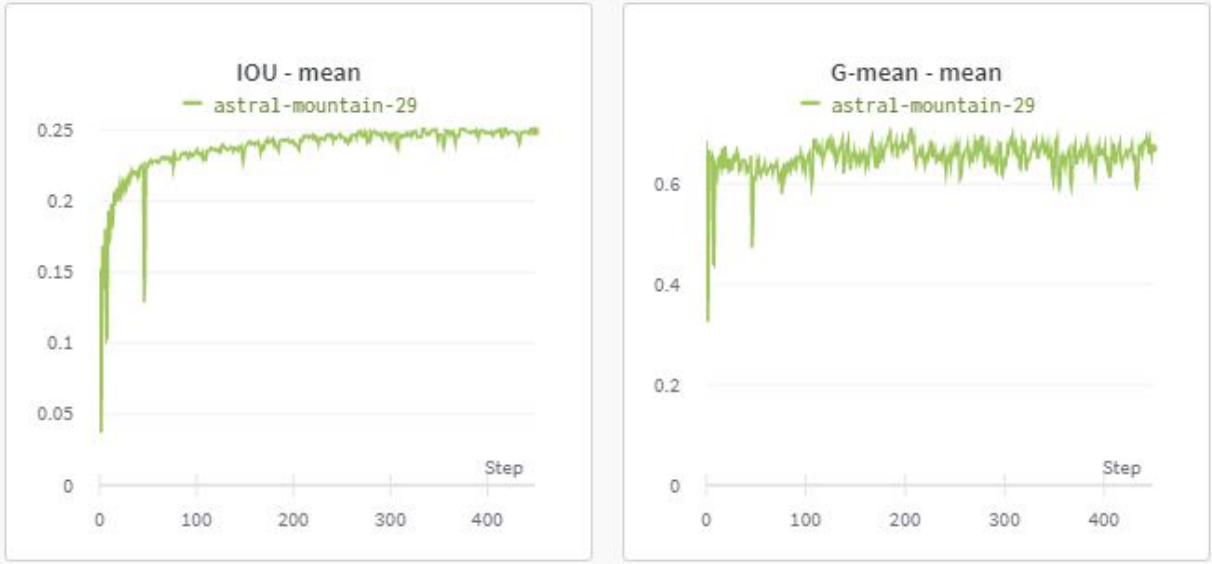

FIGURE 6 – Le IOU (0.249) et le G-mean (0.671) obtenus après 440 epochs

4.1.2 Résultats du deuxième modèle

La valeur de Dice obtenue (39.7%) est largement inférieur à celui de l'article (70.77%), nous avons essayé de modifier les paramètres du modèle avec : learning rate = 0.0003 , batch size = 2 et 600 epochs (300 epochs par stage). Nous avons obtenu les résultats suivants :

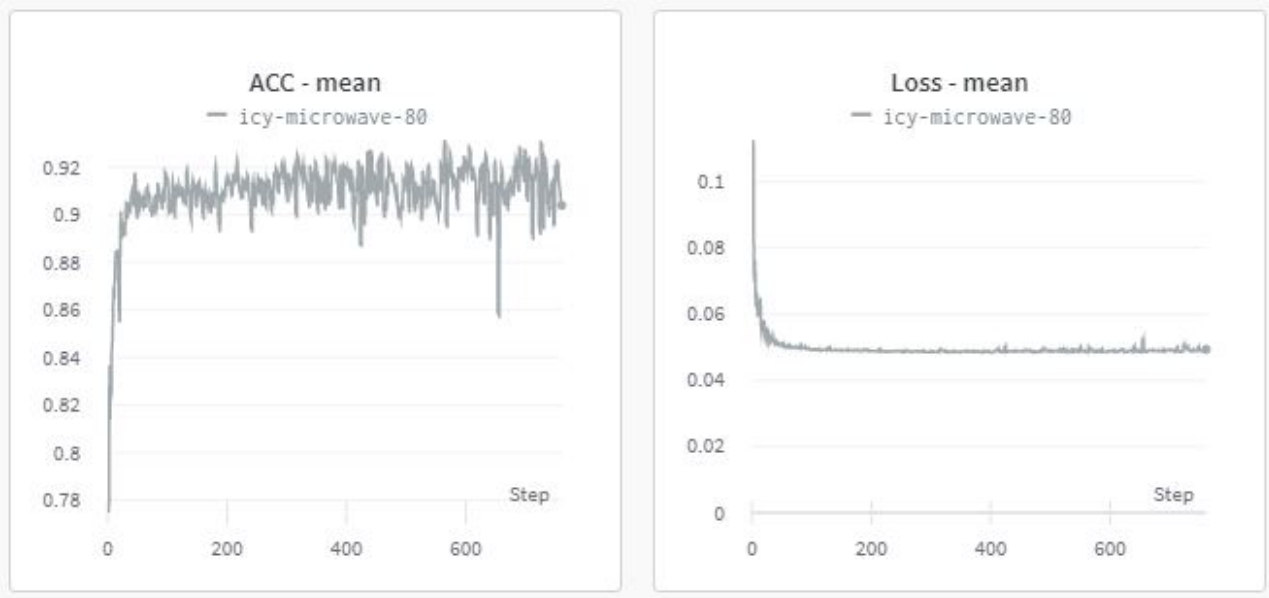

FIGURE 7 – L'accuracy (0.904) et le loss (0.0493) obtenus après 600 epochs

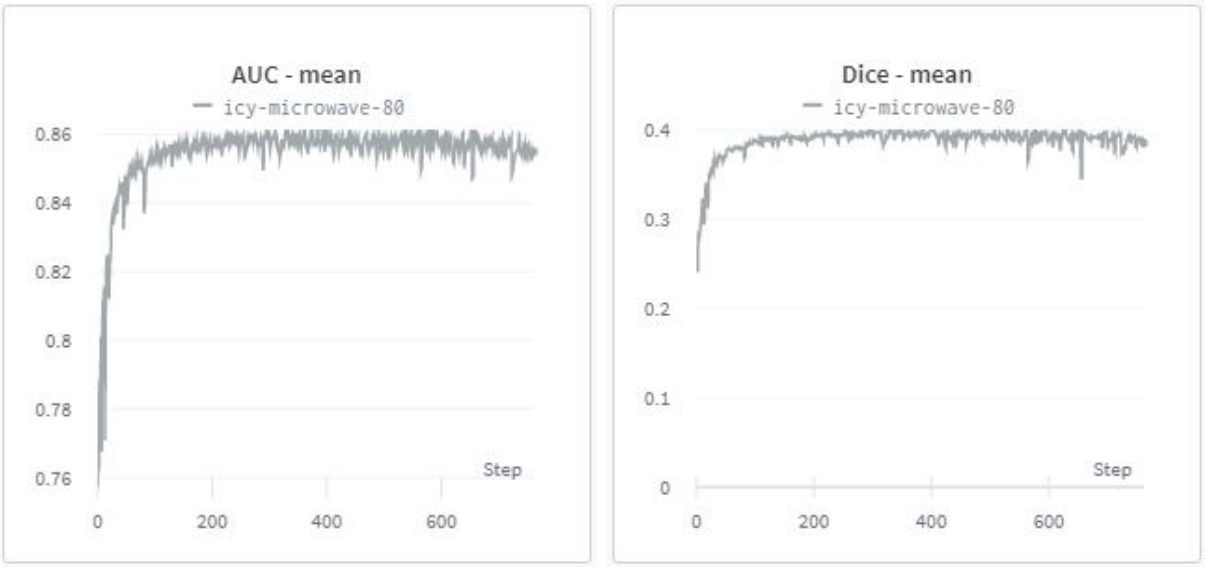

FIGURE 8 – L'AUC (0.8549) le Dice (0.3854) obtenus après 600 epochs

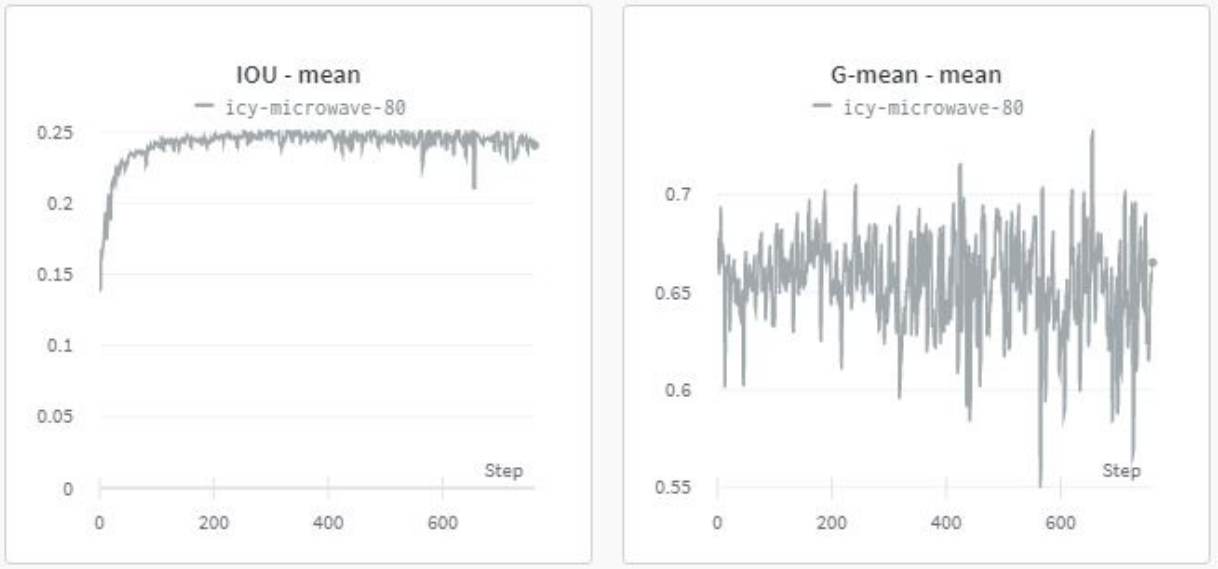

FIGURE 9 – Le IOU (0.2406) et le G-mean (0.6651) obtenus après 600 epochs

Les résultats obtenus par le premier modèle sont meilleurs que ceux du deuxième, c'est pour ça que nous avons fait les prédictions en utilisant le premier modèle. La prédiction a été effectuée sur les 22 images de data set de test de ROSE-2.

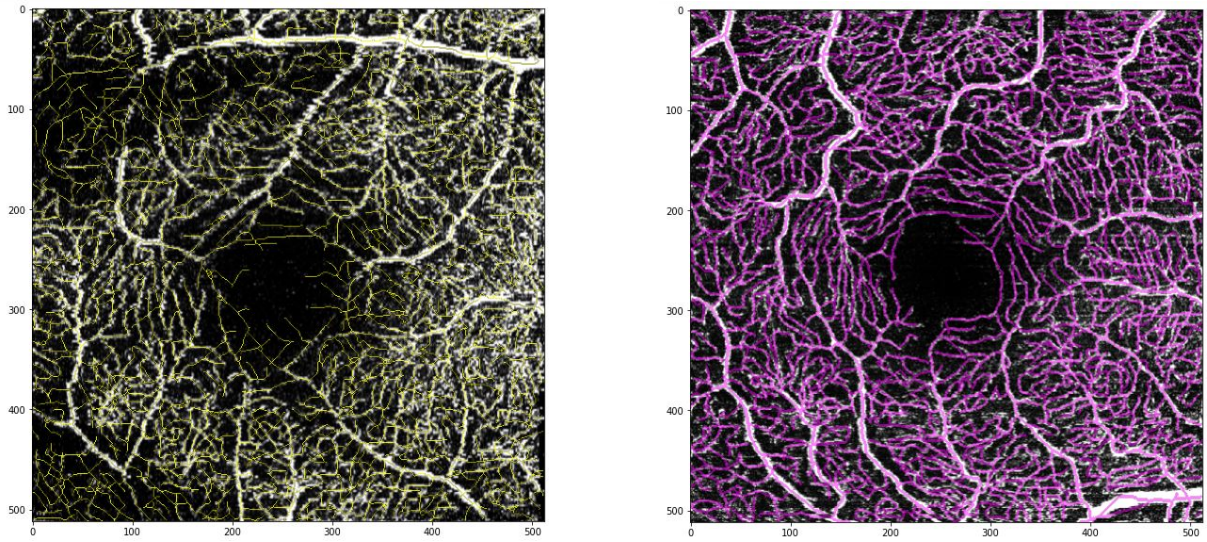

FIGURE 10 – La superposition de la vérité-terrain (vaisseaux jaunes et violets) au-dessus des images OCTA origines

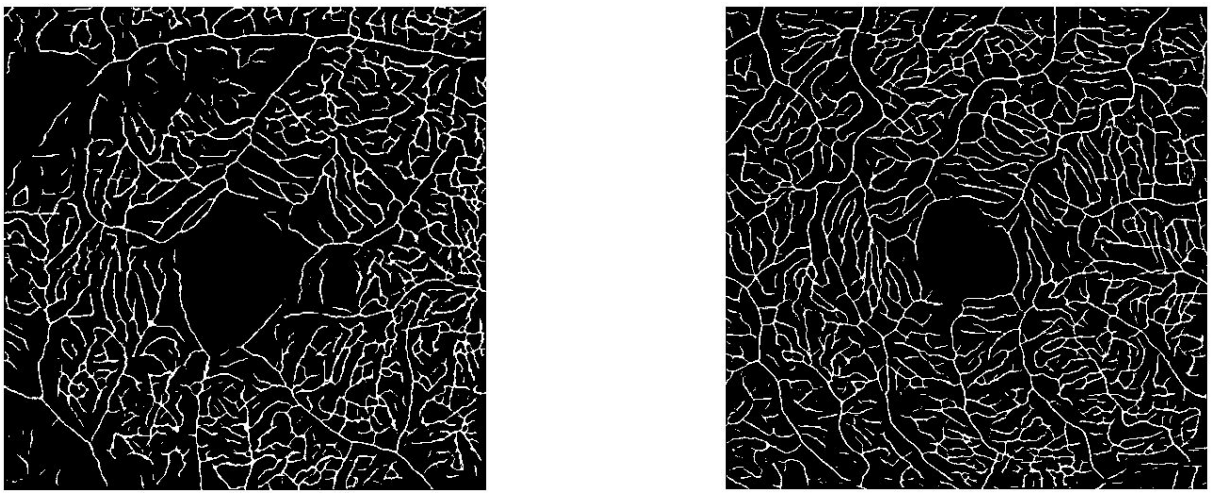

FIGURE 11 – La prédiction sur les mêmes images en utilisant le deuxième modèle.

4.1.3 La prédiction du modèle sur les données de BACLESSE :

Après avoir adapté les images OCT de BACLESSE pour les entrées de notre modèle en ajustant leur taille à [1, 3, 512, 512], nous avons passé les images à travers les étapes de prédiction de coarse et de fusion. Le résultat final est une image de segmentation sans seuillage avec des pixels variant entre 0.86451596 et 0.003775701. Nous avons trouvé une valeur optimale de seuil égale à 0.71 pour la prédiction finale.

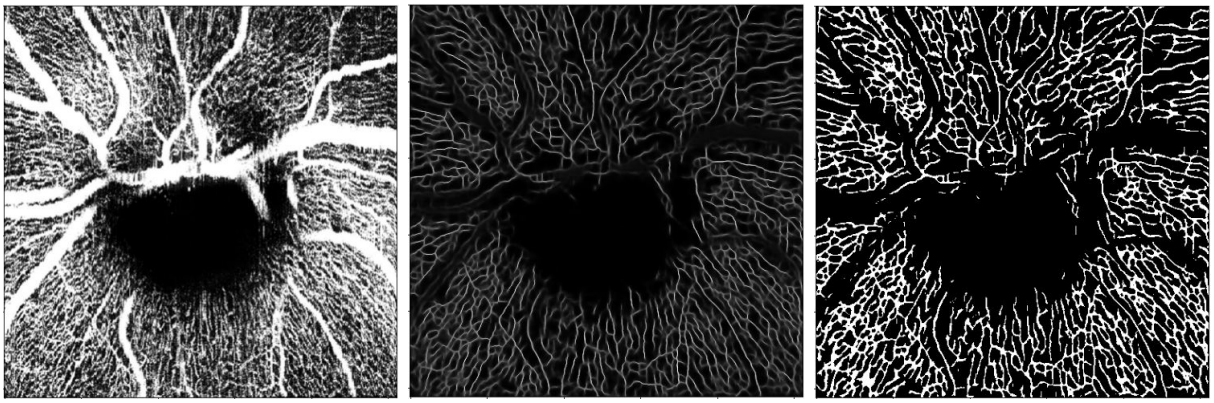

FIGURE 12 – À gauche, l'image d'origine ; au milieu, la segmentation prédite avec sans seuillage ; à droite, la segmentation prédite avec seuillage.

4.1.4 La prédiction du modèle sur les données de BACLESSE :

FIGURE 13 – À gauche, l'image d'origine ; au milieu, la segmentation prédite avec sans seuillage ; à droite, la segmentation prédite avec seuillage.

Nous constatons que les segmentations des vaisseaux sont satisfaisantes dans l'ensemble, mais que les gros vaisseaux sont plus ou moins masqués.

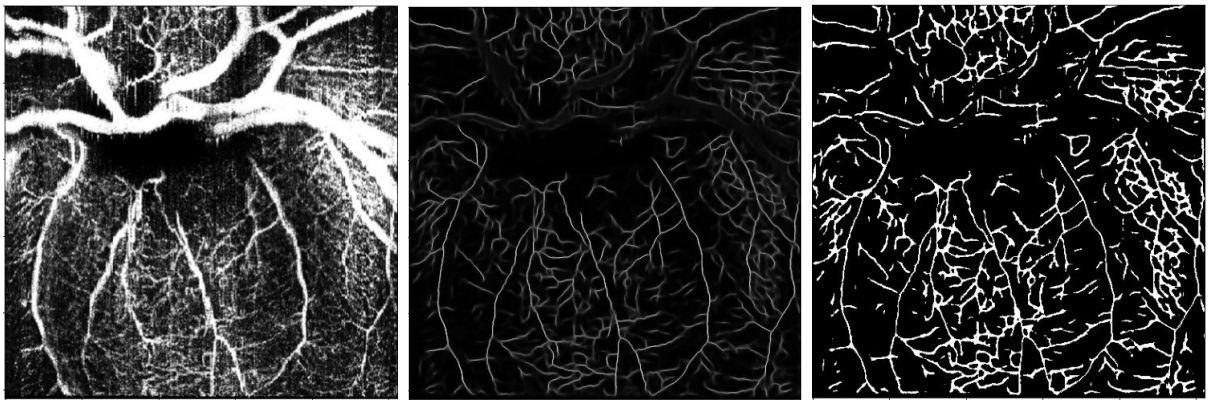

FIGURE 14 – À gauche, l'image d'origine ; au milieu, la segmentation prédite avec sans seuillage ; à droite, la segmentation prédite avec seuillage.

4.2 Algorithme VAFF-Net

Nous avons effectué des tests sur la base de données ROSE-O, la même que celle utilisée dans le travail original. Les résultats obtenus dans le travail original sur ROSE-O sont :

- pour la segmentation des vaisseaux, un score de Dice de 76.58% et un score BACC de 84.62% ;
- pour la segmentation de la FAZ, un score de Dice de 95.19% et un score BACC de 96.67%.

Nous espérons donc obtenir des résultats similaires en lançant cet algorithme.

4.2.1 Implémentation pytorch lightning sur 1000 epochs

Nous allons présenter les résultats obtenus en lançant l'algorithme sur 1000 epochs, avec notre implémentation en pytorch lightning. Les paramètres d'initialisation sont les suivants :

- learning rate = 1e-5 ;
- weight decay = 1e-5 ;
- batch size = 1 ;

A l'exception du batch size (4 dans les tests réalisés dans le papier), tous les paramètres sont les mêmes. Pour les loss, nous obtenons les courbes suivantes :

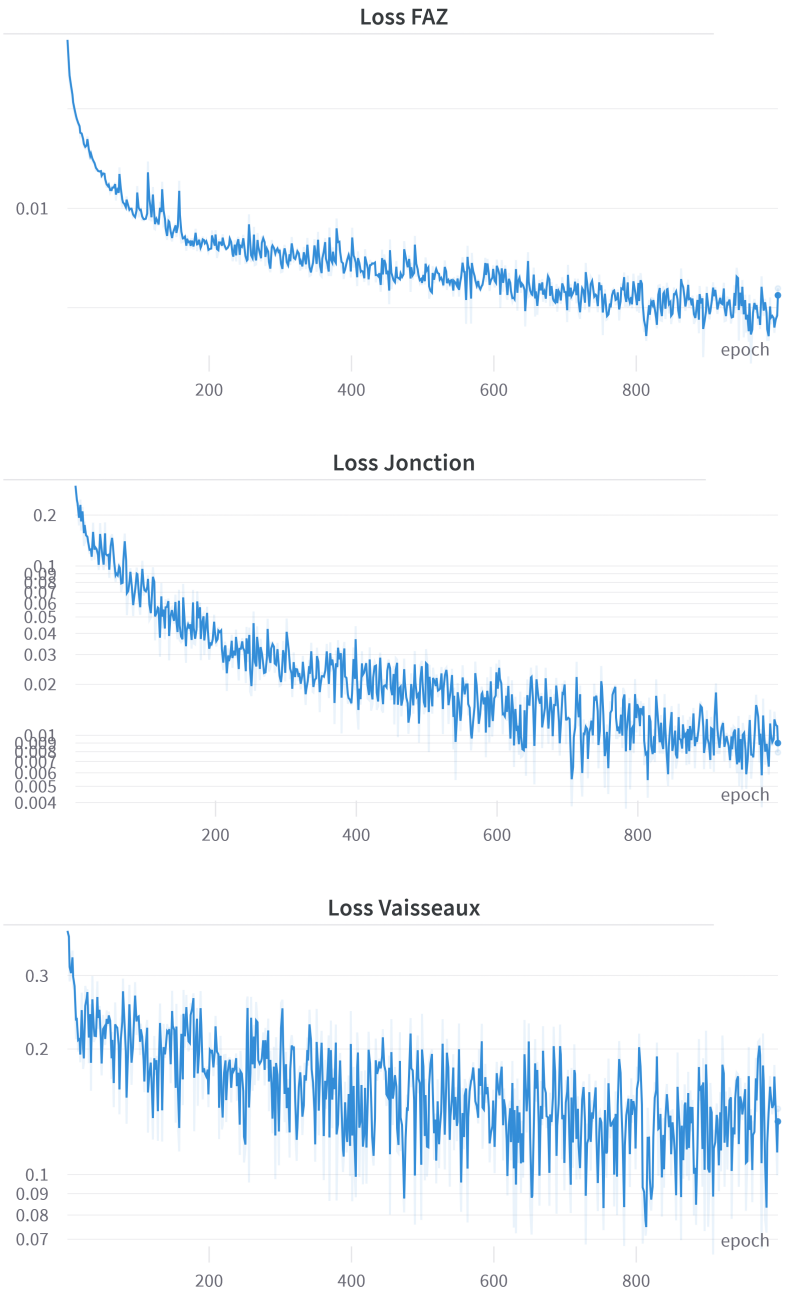

FIGURE 15 – Loss des vaisseaux, jonctions et FAZ au cours des 1000 epochs

La courbe de loss de la FAZ est correcte : elle décroît rapidement au départ, et continue à décroître plus lentement à partir de 200 epochs. La courbe de loss des jonctions n'est pas aussi correcte, elle subit plus d'à-coups et elle à du mal à atteindre une petite valeur. La courbe de loss des vaisseaux n'est vraiment pas satisfaisante ; elle est beaucoup trop instable et n'arrive même pas à descendre en-dessous de 0.1 de manière permanente.

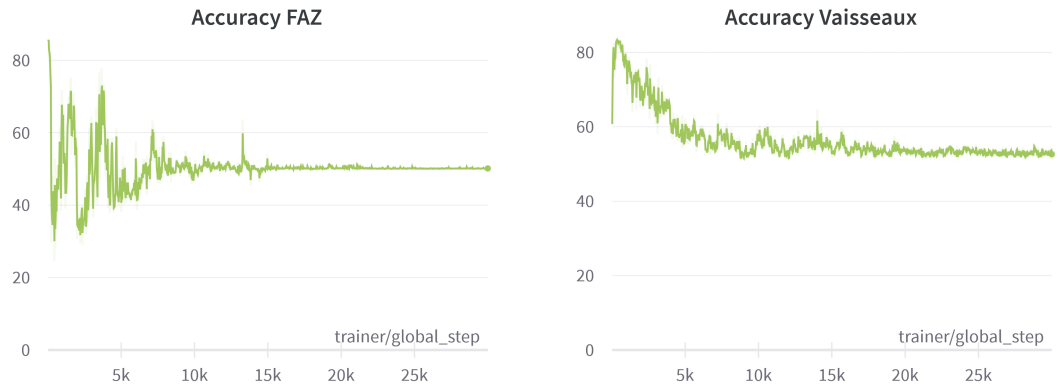

FIGURE 16 – Accuracy de la FAZ et des vaisseaux sur 1000 epochs

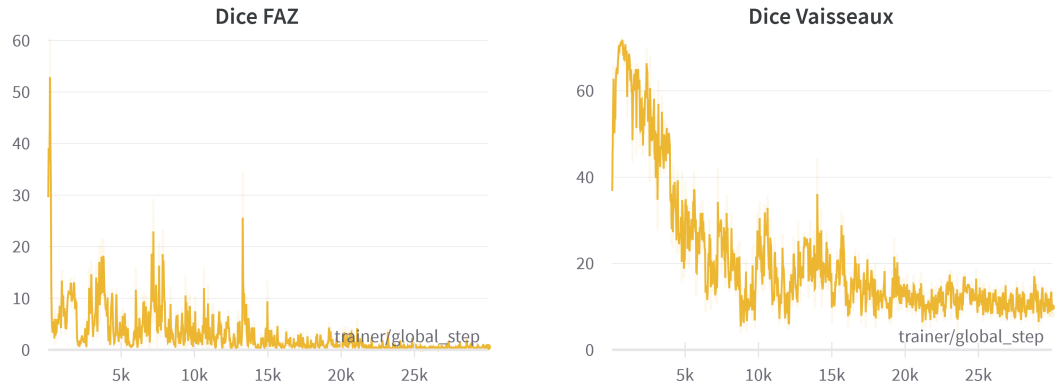

FIGURE 17 – Score de Dice de la FAZ et des vaisseaux sur 1000 epochs

Si l'entraînement s'était bien passé, nous aurions des courbes augmentant vite lors des premiers epochs puis se stabilisant pour augmenter lentement ou atteindre un plateau ; aucune de ces courbes ne suit ce schéma. Les courbes d'accuracy augmentent rapidement lors des premiers epochs mais baissent pour stagner autour de 50%. Pire, les courbes de Dice s'écroulent et finissent avec des scores très mauvais. Cela se ressent lorsqu'on regarde les prédictions sur les images de test (ici, l'image 4 de la base de test de ROSE-O) :

FIGURE 18 – À gauche, l'image d'origine et la segmentation de vaisseaux prédite en vert ; au milieu, la segmentation prédite avec thresholding ; à droite, la segmentation prédite sans thresholding.

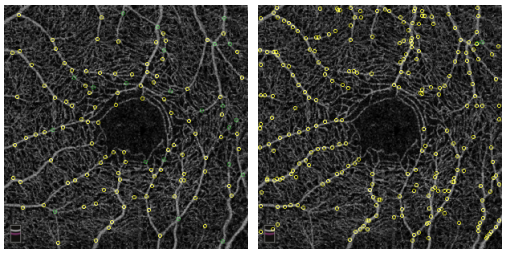

FIGURE 19 – À gauche, l'image de vérité terrain pour les jonctions entre vaisseaux ; à droite, l'image prédite.

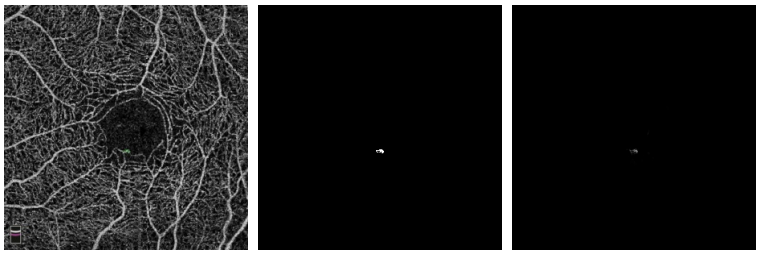

FIGURE 20 – À gauche, l'image d'origine et la segmentation du FAZ prédite en vert ; au milieu, la segmentation prédite avec thresholding ; à droite, la segmentation prédite sans thresholding.

4.2.2 Implémentation lightning avec changements de paramètres

Pour essayer d'obtenir un modèle un peu plus performant, nous avons essayé de modifier quelques paramètres :

— learning rate = 5e-4 et weight decay = 5e-5 pour le premier ;
— learning rate = 1e-4 et weight decay = 5e-5 pour le premier ;
— learning rate = 5e-4 et weight decay = 5e-4 pour le premier ;
— learning rate = 5e-4 et weight decay = 5e-5 pour le quatrième et dernier.

Voici les résultats obtenus sur ces quatre sets de paramètres :

FIGURE 21 – Scores d'accuracy et de Dice sur les vaisseaux et la FAZ sur les 4 modèles testés (50 epochs).

Le premier modèle obtient des scores corrects sur la segmentation de vaisseaux (un score de Dice de 72.69% après 50 epochs sur les exemples de test). Nous avons aussi essayé de relancer cet algorithme sur un nombre d'epochs plus importants, mais nous obtenions un plus mauvais résultat sur la segmentation des vaisseaux. Nous pouvons maintenant observer un exemple d'image prédite (la même que dans la partie précédente) pour voir si l'on obtient des bons résultats :

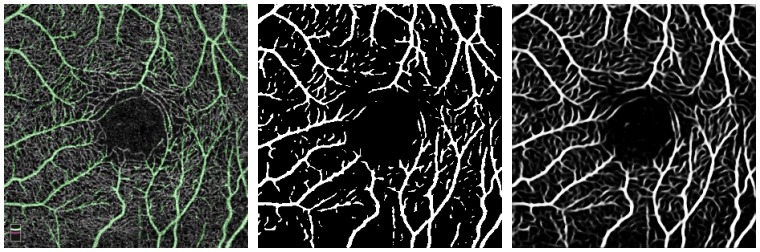

FIGURE 22 – À gauche, l'image d'origine et la segmentation de vaisseaux prédite en vert ; au milieu, la segmentation prédite avec thresholding ; à droite, la segmentation prédite sans thresholding.

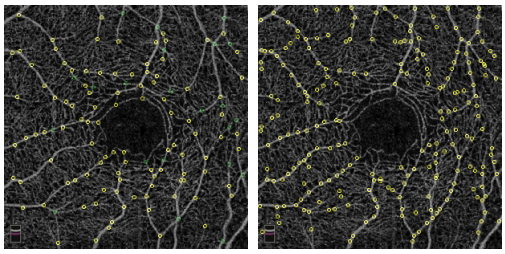

FIGURE 23 – À gauche, l'image de vérité terrain pour les jonctions entre vaisseaux ; à droite, l'image prédite.

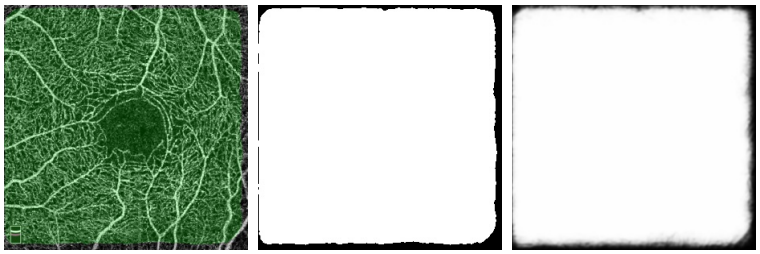

FIGURE 24 – À gauche, l'image d'origine et la segmentation du FAZ prédite en vert ; au milieu, la segmentation prédite avec thresholding ; à droite, la segmentation prédite sans thresholding.

Cette fois-ci, la segmentation des vaisseaux est plutôt satisfaisante, mais celle de la FAZ n'est pas bonne.

Nous avons chargé ce modèle pour faire de la segmentation de vaisseaux sur une image qu'il ne connaît pas. Voici un exemple sur l'image OCTa "BX010" transmise par le centre François Baclesse :

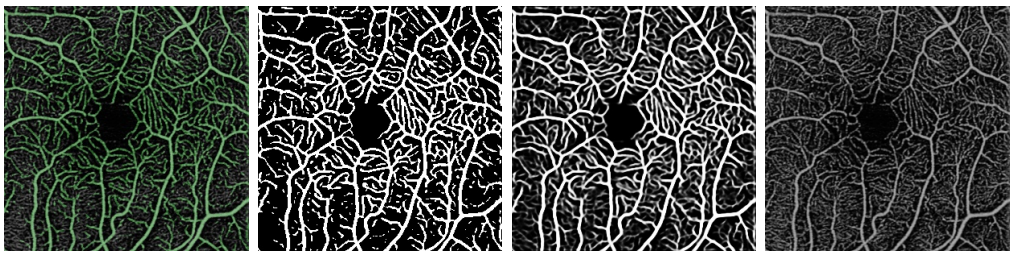

FIGURE 25 – De gauche à droite, l'image d'origine et la segmentation de vaisseaux prédite en vert, la segmentation prédite avec thresholding, la segmentation prédite sans thresholding, et l'image d'origine

Les résultats ici sont satisfaisants.

Cependant, bien qu'en utilisant le même algorithme et les mêmes données d'entraînement et de test, nous n'arrivons malheureusement pas à obtenir les bons résultats obtenus par les auteurs de cet algorithme.

5 Conclusion

5.1 Ressenti sur le projet et difficultés rencontrées

Ce projet nous a permis d'acquérir de l'expérience en nous entraînant à faire un état de l'art, utiliser et s'approprier de nouveaux outils (comme Google Colab et WandB), et utiliser et tester des algorithmes de deep-learning développé pour résoudre un problème concret.

Cependant, les nombreux ralentissements dus aux multiples problèmes rencontrés lors de l'utilisation des algorithmes nous ont posé problème, et nous n'avons pas pu avancer comme nous l'aurions souhaité. Nous n'avons pas pu atteindre les mêmes résultats que ceux annoncés par les auteurs des algorithmes que nous avons choisi (OCTA-Net et VAFF-Net), ce qui est décevant. Mais nous avons tout de même réussi à obtenir des modèles pouvant segmenter des images OCTa de manière correcte.

5.2 Pistes d'amélioration

— Avoir du temps supplémentaire, pour effectuer plus de tests et comprendre d'où vient les dysfonctionnements des algorithmes que nous avons utilisés ;
— Si nous arrivions à résoudre ces problèmes, nous pourrions essayer de développer une solution afin de corréler les images d'OCTa segmentées et la vision d'un patient.
— Une autre piste serait d'utiliser un autre type de données (par exemple, des potentiels visuels) et donc d'autres méthodes ou algorithmes pour répondre au problème de corrélation entre les données et la vision d'un patient.

Références

[1] Wikipédia, "Réseau neuronal convolutif — wikipédia, l'encyclopédie libre," 2022. [En ligne ; Page disponible le 8-juin-2022].

[2] Wikipédia, "U-net — wikipédia, l'encyclopédie libre," 2023. [En ligne ; Page disponible le 16-janvier-2023].

[3] Y. Ma, H. Hao, J. Xie, H. Fu, J. Zhang, J. Yang, Z. Wang, J. Liu, Y. Zheng, and Y. Zhao, "Rose : A retinal oct-angiography vessel segmentation dataset and new model," *IEEE Transactions on Medical Imaging*, vol. 40, no. 3, pp. 928–939, 2021.

[4] J. Hao, T. Shen, X. Zhu, Y. Liu, A. Behera, D. Zhang, B. Chen, J. Liu, J. Zhang, and Y. Zhao, "Retinal structure detection in octa image via voting-based multitask learning," *IEEE Transactions on Medical Imaging*, vol. 41, no. 12, pp. 3969–3980, 2022.

[5] M. Li, Y. Chen, Z. Ji, K. Xie, S. Yuan, Q. Chen, and S. Li, "Image projection network : 3d to 2d image segmentation in octa images," *IEEE Transactions on Medical Imaging*, vol. 39, no. 11, pp. 3343–3354, 2020.

[6] M. Li, K. Huang, Q. Xu, J. Yang, Y. Zhang, Z. Ji, K. Xie, S. Yuan, Q. Liu, and Q. Chen, "Octa-500 : A retinal dataset for optical coherence tomography angiography study," 2020.

[7] M. J. Menten, J. C. Paetzold, A. Dima, B. H. Menze, B. Knier, and D. Rueckert, "Physiology-based simulation of the retinal vasculature enables annotation-free segmentation of oct angiographs," 2022.

[8] K. M. Meiburger, M. Salvi, G. Rotunno, W. Drexler, and M. Liu, "Automatic segmentation and classification methods using optical coherence tomography angiography (octa) : A review and handbook," *Applied Sciences*, vol. 11, no. 20, 2021.

[9] "Rose12 - imed.nimte.ac.cn." `https://imed.nimte.ac.cn/dataofrose.html`. [En ligne ; accès le 26-Janvier-2023].

[10] "Rose-o - imed.nimte.ac.cn." `https://imed.nimte.ac.cn/ROSE-O.html`. [En ligne ; accès le 26-Janvier-2023].

[11] "Octa-500 | ieee dataport." `https://ieee-dataport.org/open-access/octa-500`. [En ligne ; accès le 26-Janvier-2023].