Segmentation des vaisseaux rétiniens à l'aide de U-net, Attention U-Net et Unet++

HAWCHAR, Mohamad
Université du Québec à Montréal
Présenté à : Dr. Joël Lefebvre

Dans le cadre du cours INF899G
Laval, Canada
hawchar.mohamad@courrier.uqam.ca

***Résumé* — La segmentation automatique des images du fond de la rétine pour extraire les vaisseaux sanguins est une tâche essentielle pour aider les médecins à diagnostiquer les maladies liées à la rétine comme l'hypertension, le glaucome et la rétinopathie diabétique, qui sont les principales causes de cécité.**

Dans ce projet, nous avons implémenté trois architectures de codage-décodage différentes : U-net [1], Attention U-Net [2] et Unet++ [4], sur la combinaison d'images de trois jeux de données différents : STARE [5], DRIVE [6] et CHASEDB1 [7].

I. Introduction

Les attributs des vaisseaux rétiniens, tels que la longueur, la largeur et le modèle de ramification et les angles, jouent un rôle important dans le diagnostic et le traitement de diverses maladies cardiovasculaires et ophtalmologiques telles que le diabète, l'hypertension et l'artériosclérose. L'étape cruciale avant l'extraction de ces caractéristiques morphologiques des images rétiniennes est la segmentation des vaisseaux.

Dans ce travail, nous proposons une méthode pour la segmentation des vaisseaux rétiniens basée sur des réseaux en forme de U-net. L'argumentation des données est appliquée en faisant tourner de 45 degrés, en inversant horizontalement et verticalement les images d'entraînement. Trois architectures de réseaux entièrement convolutifs, U-Net [1], Attention U-Net [2] et Unet++[4] : Nested Unets, sont utilisées pour la segmentation des vaisseaux. Les performances de notre méthode sont évaluées sur trois jeux de données publics : DRIVE [18], STARE [10] et CHASE DB1 [20]. Nous utilisons la précision et l'aire sous la courbe ROC comme principales mesures d'évaluation, en plus du score F1, de la précision et de l'accuracy. Dans ce rapport, nous allons examiner en détail le prétraitement, l'augmentation, la méthodologie et les résultats de notre expérience.

II. Théorie

La segmentation d'images est l'un des sujets fondamentaux de la vision par ordinateur. Les architectures de réseaux neuronaux encodeur-décodeur en forme de U peuvent apprendre automatiquement une hiérarchie de plus en plus complexe de caractéristiques directement à partir des données d'entrée. Elles présentent des améliorations frappantes par rapport aux systèmes reposant sur des méthodes de traitement d'images antérieures.

Dans ce travail, nous avons partiellement appliqué l'architecture U-net standard pour servir comme algorithme de mesure et de comparaison, et nous avons appliqué deux autres architectures : l'U-net d'attention qui essaie d'améliorer l'U-net standard en appliquant une fonction d'attention sur la porte de chaque décodeur pour l'aider à se concentrer sur les informations importantes, la troisième architecture est l'Unet++ ou l'Unet imbriqué qui fonctionne en concaténant beaucoup plus d'informations pour la partie de connexion de saut.

III. Données

J'ai utilisé trois différents ensembles de données accessibles au public pour entraîner et évaluer la méthode proposée : DRIVE [5], STARE [6], et CHASE DB1 [7]. Un ensemble de segmentations manuelles est disponible dans ces trois jeux de données, que nous utilisons comme base.

Le jeu de données DRIVE est composé de 40 images, et chaque image a une résolution de 565×584, nous avons décidé d'en utiliser 20 pour la formation et 20 pour la validation et le test. Le jeu de données STARE fournit 20 images d'une résolution de 700×605, nous en avons utilisé 15 pour l'apprentissage et 5 pour la validation. Enfin, le jeu de données ChaseDB1 fournit 28 images de résolution 999×960, nous en avons utilisé 20 pour l'apprentissage et 8 pour la validation. Nous avons donc obtenu 55 images pour l'apprentissage et 39 pour la validation et le test.

A. Prétraitement

Le prétraitement est une étape essentielle dans cette mise en œuvre, il vise à rendre les vaisseaux plus visibles et à favoriser

leur formation. Il y a quatre étapes dans le prétraitement des images originales : la conversion de l'échelle de gris, la normalisation, l'égalisation adaptative d'histogramme limitée par le contraste (CLAHE) et l'ajustement gamma. [3]

1. *Conversion en échelle de gris*

The colors are not essential for this task, this is why we convert all the images to grey using the next function:

$$I = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

2. *Normalisation Z-score*

La normalisation du z-score est appliquée pour chaque pixel de toutes les images afin d'accélérer le taux d'apprentissage, en utilisant l'équation suivante :

$$\acute{x} = \frac{x_i - \bar{X}}{\sigma_X}$$

3. *L'égalisation adaptative d'histogramme limitée par le contraste (CLAHE)*

CLAHE a pour but d'améliorer le contraste de l'image en redistibuant la lumière après le calcul de plusieurs histogrammes, pour l'implémenter nous avons utilisé la fonction intégrée dans openCV.

4. *Réglage du gamma*

Nous l'avons utilisé spécifiquement pour améliorer le contraste dans les zones de faible intensité, c'est-à-dire pour faire apparaître plus clairement le vaisseau dans les zones sombres.

$$J = 255(\frac{I}{255})^{1/\gamma}$$

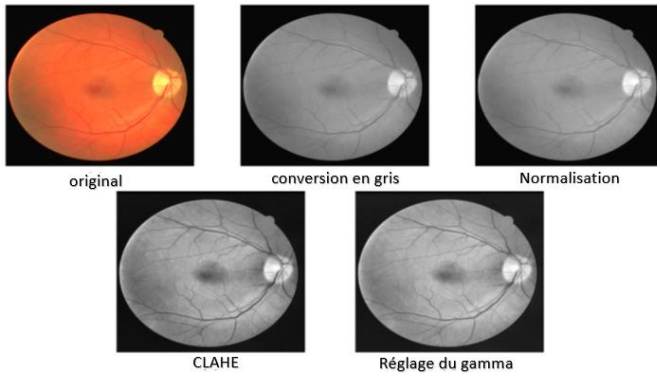

Figure 1: Résultats du prétraitement après chaque étape : Image originale de la rétine. Conversion en échelle de gris. Normalisation. CLAHE. Ajustement gamma.

B. Augmentation

En raison du nombre limité d'images dont nous disposons (40 dans DRIVE, 20 dans STARE et 28 dans CHASE DB1), l'augmentation joue un rôle très important pour assurer une bonne généralisation lorsque nous n'avons pas assez de données. Nous avons donc appliqué trois types d'augmentation des données d'image : Retournement horizontal, retournement vertical et rotation de 45 degrés.

Ce processus nous a permis d'augmenter nos données d'entraînement de 55 à 220 images avec leurs masques, ce qui s'est avéré suffisant pour une bonne généralisation après l'étape de test.

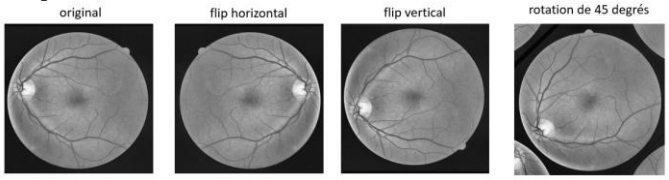

Figure 2: Exemple des résultats des trois méthodes d'augmentation d'image sur la même image.

IV. Méthodologie

Dans ce projet, j'ai mis en œuvre trois différentes architectures Unet (U-net, Attention U-net et Unet++ : Nested Unet) pour évaluer et voir la différence dans les résultats de chacune d'entre elles.

A. U-net

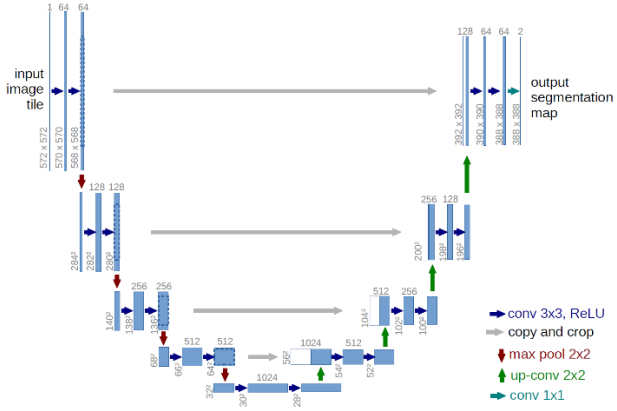

Figure 3: L'architecture originale de U-net. Elle se compose de 4 couches de codage, de 4 couches de décodage, d'une couche de goulot d'étranglement et de connexions de saut. [1]

L'architecture U-net se compose d'un codeur et d'un décodeur qui sont connectés par une couche d'étranglement et des couches de connexion de saut. Nous avons quatre blocs Encoder et Decoder, et un bloc bottlneck. Chaque bloc fait 2 convolutions puis effectue un suréchantillonnage ou un sous-échantillonnage en fonction du type de couche. Au niveau de la couche décodeur, nous concaténons la connexion de saut avec le résultat du suréchantillonnage pour essayer de sauvegarder les données spatiales de l'image.

B. Attention U-Net

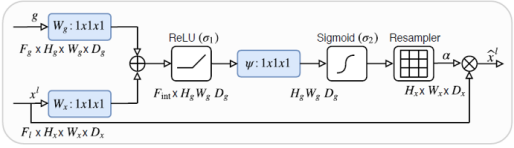

Figure 4: La fonction d'attention sur laquelle repose l'Attention U-net. [2]

La seule différence entre le U-net standard et le U-net d'attention est que ce dernier utilise une fonction d'attention que nous pouvons voir dans la figure 4. La fonction d'attention est appliquée dans chaque couche du décodeur avant la conctenation de la connexion de saut avec la dernière convolution suréchantillonnée. La variable x représente la connexion de saut et g la convolution suréchantillonnée, nous commençons par appliquer un stride (1×1) sur g, et un stride (2×2) sur x pour leur donner la même taille. Ensuite, nous concaténons les deux et nous appliquons une fonction relu qui élimine les valeurs négatives, à ce stade nous obtenons une matrice à une dimension avec la même hauteur et largeur que g qui représente les poids d'importance pour chaque pixel. Nous terminons en appliquant une fonction sigmoïde pour que les valeurs soient comprises entre 0 et 1, et nous rééchantillonnons l'image à la même taille que la connexion sautée, et nous multiplions les résultats avec le x original pour obtenir le résultat de la connexion sautée pondérée.

C. *Unet++: Nested U-Net*

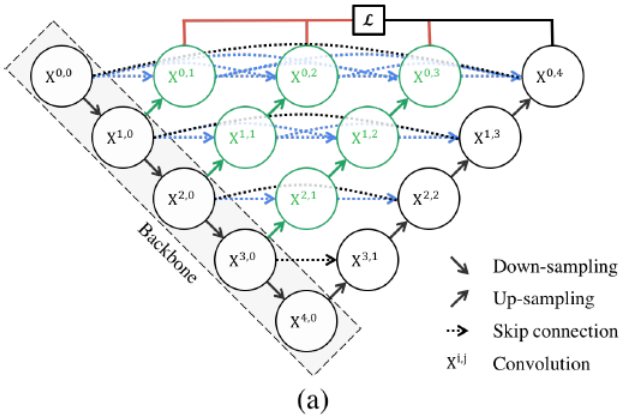

Figure 5: L'architecture de l'Unet++ : U-Net imbriqué. Chaque nœud autre que la couche dorsale est le résultat du sur-échantillonnage du nœud précédent, et de la concaténation avec tous les nœuds horizontaux précédents. Exemple : pour obtenir X0,2, nous sur-échantillonnons X1,1 et le concaténons avec X0,1 et X0,0. [4]

Cette architecture vise à sauvegarder autant d'informations que possible des nœuds précédemment calculés et à les utiliser dans le calcul du nouveau nœud. Tous les noeuds autres que les noeuds de la couche Backnone sont le résultat de la conatination de la version upsamples du noeud calculé précédemment, et de tous les noeuds qui existent horizontalement sur la même ligne que le nouveau noeud. Par conséquent, le noeud de sortie X0,4 est le résultat de la concaténation de la version suréchantillonnée de X1,3 avec X0,3 X0,2 X0,1 et X0,0.

IV. Resultats & Experimentation

L'entraînement et le test des réseaux proposés ont été réalisés à l'aide de Pytorch sur une machine Google Colab pro.

Nous commençons par fixer les hyperparamètres pour toutes les architectures, car le but de ce projet est de comparer les résultats des trois architectures proposées. Nous avons fixé la taille des images à 512×512 pour les trois jeux de données utilisés. Nous avons choisi un taux d'apprentissage de 0.0001 et la fonction de perte Dice, que nous avons trouvé très populaire pour les architectures U-net. Nous avons également fixé la taille du lot à 2 images, et le nombre d'époques à 50 afin d'obtenir le meilleur résultat possible pour les trois architectures. Enfin, nous avons utilisé l'optimiseur ADAM.

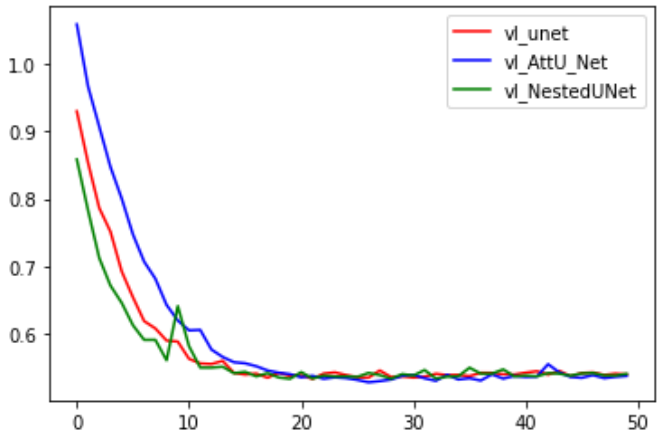

Figure 6: Comparaison des pertes de validation entre les trois architectures proposées (rouge pour U0net, bleu pour Attention U-net et vert pour Nested Unet++).

Nous remarquons dans la Figure 6 que les résultats de la perte de validation sont assez similaires entre les trois architectures, la seule différence est que Unet++ : Nested Unet converge le plus rapidement, alors que l'Attention U-net converge le plus lentement. Nous remarquons également que les trois architectures atteignent leur apprentissage maximum à l'époque entre 20 et 25. La raison pour laquelle l'Unet++ converge le plus vite est qu'il est celui qui sauvegarde le plus d'informations des étapes précédentes. De même, l'U-net d'attention converge le plus lentement parce qu'il a besoin de temps (itérations d'époques) pour apprendre exactement où chercher les navires, et concentrer l'attention sur eux. Enfin, nous devons noter que le temps de formation pour l'Unet++ est trois fois plus long que celui de l'Attention U-net et cinq fois plus long que celui de l'U-net standard.

Tableau d'évaluation

	Perte minimale	***Score F1***	***Précision***	***Accuracy***	***ROC***
U-net	0.5328	0.8136	0.7755	0.9690	**0.9125**
Attention U-Net	**0.5278**	**0.8194**	0.8026	**0.9708**	0.9049
Unet++	0.5329	0.8152	**0.8034**	0.9704	0.8988

Figure 7: Tableau d'évaluation des trois méthodes proposées, selon les 5 métriques Perte minimale, Score F1, Précision, Accuracy et ROC.

Le tableau de la figure 7 nous montre que les résultats des trois modèles son

t relativement proches, le meilleur résultat pour le score F1 est celui de l'Attention U-net avec 0,8194, pour la précision c'est l'Unet++ avec 0,8034, et pour l'exactitude c'est encore l'Attention U-net qui l'emporte avec 0,9708. Le meilleur score pour le ROC AUC est pour le U-net avec 0.9125. Nous pouvons conclure que les modèles Unet++ et Attention U-net sont plus performants que le modèle U-net standard, et que leurs résultats sont relativement proches les uns des autres. Étant donné que le modèle Attention U-net est plus rapide à former et plus simple du point de vue de l'architecture, et qu'il a donné le meilleur résultat pour la plupart des mesures, il est sûr de recommander le modèle Attention U-net pour le problème de la segmentation des vaisseaux rétiniens.

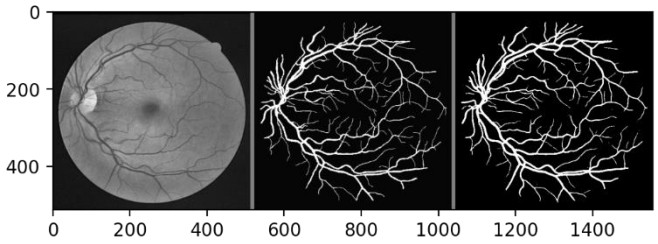

Figure 8:Résultats de l'architecture Attention U-net : de gauche à droite : Original, vérité terrain, résultat de la segmentation.

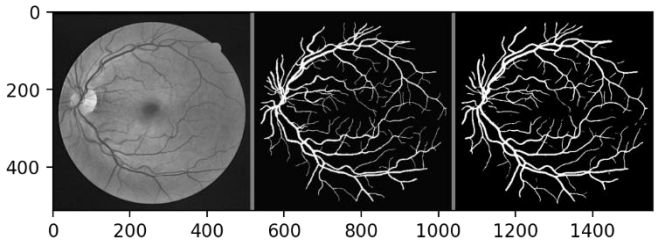

Figure 9: Résultats de l'architecture Unet++ : de gauche à droite : Original, vérité terrain, résultat de la segmentation

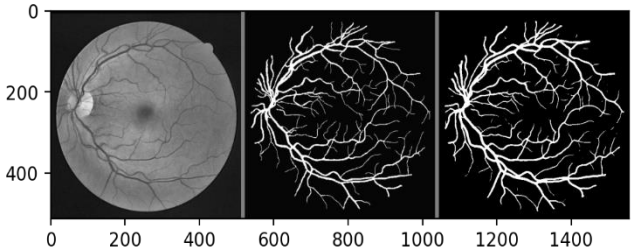

Figure 10: Résultats de l'architecture U-net : de gauche à droite : Original, vérité terrain, résultat de la segmentation.

Discussion & Conclusion

Dans ce projet, nous avons utilisé trois ensembles de données différents de segmentation rétinienne, nous avons appliqué des prétraitements pour rendre les vaisseaux plus visibles et faciliter l'apprentissage de l'algorithme. Nous avons également appliqué trois méthodes d'augmentation des données pour résoudre le problème des données limitées dont nous disposons. Ensuite, nous avons appliqué trois architectures différentes en forme de U-net et comparé leurs résultats sur différentes métriques.
Nous avons conclu que les performances des deux architectures Attention U-net et Unet++ donnaient des résultats proches, l'Attention U-net dépassant légèrement l'Unet++ sur la plupart des mesures.
Pour améliorer la performance de cette tâche, nous suggérons de faire une extraction de patchs et d'alimenter les réseaux avec les images en patchs, ce sera une bonne méthode d'augmentation qui nous aidera avec le manque de données que nous avons rencontré. Nous suggérons également d'expérimenter des architectures plus complexes comme DenseNet ou DeepLabv3, ou d'essayer de développer une méthode de génération d'images pour résoudre cette tâche.

References

[1] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

[2] Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., ... & Rueckert, D. (2018). Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*.

[3] Liu, Z. (2019). Retinal vessel segmentation based on fully convolutional networks. *arXiv preprint arXiv:1911.09915*.

[4] Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support* (pp. 3-11). Springer, Cham.

[5] Niemeijer, Meindert, Joes Staal, Bram van Ginneken, Marco Loog, and Michael D. Abramoff. "Comparative study of retinal vessel segmentation methods on a new publicly available database." In *Medical imaging 2004: image processing*, vol. 5370, pp. 648-656. International Society for Optics and Photonics, 2004.

[6] Hoover, A. D., Kouznetsova, V., & Goldbaum, M. (2000). Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. *IEEE Transactions on Medical imaging*, *19*(3), 203-210.

[7] Owen, C. G., Rudnicka, A. R., Mullen, R., Barman, S. A., Monekosso, D., Whincup, P. H., ... & Paterson, C. (2009). Measuring retinal vessel tortuosity in 10-year-old children: validation of the computer-assisted image analysis of the retina (CAIAR) program. Investigative ophthalmology & visual science, 50(5), 2004-2010.