Learning Active Contour Models for Medical Image Segmentation

Nissim Maruani

18 avril 2022

1 Étude synthétique de l'article

1.1 Problème traité

L'article étudié [CWV+19] s'attaque au problème de segmentation dans une image, appliqué ici au milieu médical. Les images concernées sont des IRM du cœur dont il faut segmenter trois régions : le ventricule droit, le ventricule gauche et le myocarde. La détection de ces régions est un enjeu fondamental puisqu'elle permet au praticien de prévenir ou d'identifier des maladies cardiovasculaires qui tuent chaque année 17.9 millions de personnes. La segmentation consiste à partir d'une image $u \in \mathbb{R}^{H\times W\times d}$ (de dimension $d = 3$ pour une image en couleur RGB, $d = 1$ pour une image en niveau de gris) à produire une image segmentée $v \in \{0...k-1\}^{H\times W}$ où chaque pixel peut appartenir à k régions différentes.

1.2 Méthodes utilisées et originalité de l'article

Pour segmenter ces trois régions, l'article propose d'utiliser des réseaux de neurones convolutifs (aussi appelés CNN). Les convolutions, invariantes par translations, sont très utilisées dans l'analyse d'images : associées à des fonctions d'activations non linéaires (ici $ReLU(x) = \max(x, 0)$) et à des sous échantillonnages à différentes échelles (MaxPool), elles permettent de modéliser efficacement des phénomènes complexes.

Pour entraîner ces réseaux de neurones, on utilise ici des images IRM segmentées précisément par des humains : on calcule l'erreur (en anglais *loss*) entre la segmentation prédite par l'algorithme et celle exacte puis l'on modifie les poids des convolutions par une descente de gradient (en anglais *back-propagation*). Des nouvelles images (qui n'ont pas été vues par le réseau pendant la phase d'entraînement) permettent de mesurer les performances de l'algorithme.

L'article étudié propose d'utiliser une structure classique de réseau convolutif appelé U-Net [RFB15]. Chacune des trois régions est segmentée par un réseau différent. L'output de chaque réseau est donc une image en niveau de gris $y_{pred} \in [0, 1]^{H\times W}$ qui nous permet d'obtenir la segmentation prédite $1_{y_{pred}>0.5} \in \{0, 1\}^{H\times W}$. Pour les problèmes de classification, la loss la plus classique est la *Cross Entropy Loss*, définie par :

$$CE(y_{pred}, y_{true}) = -\frac{1}{H \times W} \sum_{i,j} y_{true}(i, j) \log\left(y_{pred}(i, j)\right) + (1 - y_{true}(i, j)) \log\left(1 - y_{pred}(i, j)\right)$$

L'originalité de l'article est de proposer une fonction de perte inspirée des contours actifs qui prend en compte non seulement le label des régions, mais aussi la longueur de la bordure, à savoir :

$$L(y_{pred}) = \sum_{i,j} \sqrt{\triangle_h\ y_{pred}(i, j)^2 + \triangle_v\ y_{pred}(i, j)^2} \text{ avec } \begin{cases} \triangle_h\ u(i, j) = u(i + 1, j) - u(i, j) \\ \triangle_v\ u(i, j) = u(i, j + 1) - u(i, j) \end{cases}$$

Pour obtenir la loss AC suivante, paramétrée par $\lambda \in \mathbb{R}$:

$$AC(y_{pred}, y_{true}) = L(y_{pred}) + \lambda \sum_{i,j} y_{true}(i, j)(1 - y_{pred}(i, j))^2 + (1 - y_{true}(i, j))y_{pred}(i, j)^2$$

Pour mesurer les performances de l'algorithme, deux métriques sont utilisées, le *Dice Score* et la distance de Hausdorff. Le calcul du *Dice Score* est le suivant :

$$DC(y_{pred}, y_{true}) = 2\frac{\sum_{i,j} y_{true}(i,j)y_{pred}(i,j)}{\sum_{i,j} y_{true}(i,j) + y_{pred}(i,j)}$$

La distance de Hausdorff mesure la distance entre deux segmentations. Dans un cadre général, pour deux ensembles de points A and B, la distance de Hausdorff est définie par :

$$H(A,B) = max(h(A,B), h(B,A)) \text{ avec}$$

$$h(A,B) = \max_{a \in A} \min_{b \in B} ||a - b||_2$$

1.3 Comparaison au cours

Nous avons vu en cours des approches classiques (sans apprentissage) permettant de segmenter des formes [BBC07] [LYC06] [CMC16] [CK97]. Ces approches fonctionnent également à l'aide de la minimisation d'une énergie : c'est d'ailleurs l'une d'entre elles [CK97] qui a inspiré la loss AC utilisée dans l'article. Ces approches classiques sont cependant plus susceptibles de rester coincées dans un minimum local. Elles sont également plus lentes, puisque la descente de gradient est faite individuellement sur chaque image. De plus, l'initialisation de ces contours actifs est un problème non trivial, et l'on souhaite ici se passer d'un input de l'utilisateur : un réseau de neurone ne requiert aucune autre information que l'image pour segmenter les sections cherchées.

1.4 Nouveaux résultats

Les résultats annoncés dans l'article surpassent l'état de l'art. Avec la nouvelle métrique, la distance de Hausdorff obtenue est 40% meilleure et le Dice Score passe de 95% à 98.6% sur ce problème de segmentation.

1.5 Avis critique

Une lecture approfondie de l'article soulève plusieurs questions. La première concerne la robustesse de la méthode : l'article prétend que le paramètre λ importe peu puisque l'on obtient les mêmes résultats quelque soit $\lambda \in [1, 50]$. Cependant un seul dataset est ici considéré. Il est donc difficile de savoir si l'on aura toujours la même tolérance sur λ et plus généralement si un algorithme entraîné sur ce dataset sera capable de traiter des images légèrement différentes (orientation différente, compression plus ou moins importante, bruit éventuel...).

Un autre point est la séparation du dataset en trois ensembles : entrainement, validation et test. Cette séparation permet de garantir que l'algorithme produit des segmentations correctes sur des nouvelles images (et pas seulement sur celles qu'il a déjà vu). En lisant l'article, on ne sait pas comment cette séparation est effectuée alors que cette question est fondamentale : comme les images sont des vues en coupe des patients, deux coupes successives sont très semblables (et très corrélées). Si les images de l'ensemble test sont choisies au hasard parmi le dataset (comme c'est souvent le cas en machine learning), il y a donc des images très similaires dans les trois ensembles et **les résultats annoncés ne peuvent être considérés comme valables**. Il est essentiel de séparer les ensembles par patient : nous vérifierons ce point dans la deuxième partie.

Nous ne comprenons pas pourquoi l'article traite chaque région une par une, au lieu de segmenter les trois à la fois (ce qui possible avec les architectures U-Net et dense U-Net). En effet, puisque rien n'interdit que les trois prédictions se recouvrent, nous pensons que la combinaison de celles-ci à posteriori induit une perte de performance. Nous reconnaissons toutefois que cette séparation permet une implémentation plus facile, à la fois pour l'entraînement et le calcul des différentes métriques.

2 Implémentation et résultats

2.1 Ressources utilisées

Dans le cadre de ce projet, nous avons utilisé Python et PyTorch. Les calculs ont été effectués en SSH sur une machine Linux disposant d'une carte graphique de 8 Go. Nous avons développé l'intégralité du code présent sur notre GitHub à l'exception de certains fichiers Python dont nous indiquons systématiquement la source. En particulier, nous avons utilisé :

— La AC loss fournie sur le GitHub associé à l'article
— Les implémentations des réseaux U-Net et Dense U-Net
— Les utilitaires nécessaires pour charger les images du dataset

Nous avons implémenté :

— L'entraînement des réseaux
— La visualisation des résultats
— Le calcul des métriques
— La gestion automatique des données (notamment afin d'évaluer la robustesse des réseaux)

2.2 Spécificité de notre implémentation

Comme expliqué dans l'article, le réseau Dense U-Net prend 8 fois plus de temps à faire une prédiction que le réseau U-Net. Après quelques expérimentations, nous avons constaté que le réseau Dense U-Net, plus lourd, nécessitait également plus d'epochs (c'est à dire de passages sur le dataset) pour converger. Nous avons donc décidé d'utiliser uniquement le réseau U-Net afin d'avoir un temps d'entraînement relativement rapide sur GPU (une dizaine de minutes).

Les différents résultats de l'article montrent que les deux métriques considérées (Dice Score et distance de Hausdorf) sont très corrélées. Nous avons donc décidé d'implémenter uniquement le Dice Score.

2.3 Méthode

Nous avons détaillé les équations utilisées dans la partie précédente, voici maintenant un résumé de la méthode. La première étape consiste à séparer le dataset en trois ensembles D_{train}, D_{test} et $D_{validation}$ de cardinal respectivement 80, 10 et 10. Pour entraîner l'algorithme, on procède de la façon suivante :

— Sélection de $i \in \{0, 1, 2\}$ (région à segmenter) et de $L \in \{AC, CE\}$ (fonction de perte).
— Création du réseau $U_{i,L}$
— Pour chaque minibatch $x \in D_{train}$:
    - Calcul de la prédiction $y_{pred} = U_{i,L}(x)$
    - Calcul de la perte $L(y_{pred}, y_{true})$
    - Ajustement des poids de $U_{i,L}$ par descente de gradient
— Enregistrement de $U_{i,L}$

Une fois nos 3 réseaux entraînés pour la perte choisie, on peut segmenter une image x très rapidement (en quelque secondes) :

— Sélection de $L \in \{AC, CE\}$ (fonction de perte).
— Création de l'image finale S.
— Pour $i \in \{1, 2, 3\}$:
    - Chargement du réseau $U_{i,L}$
    - Création d'un masque $m = (U_{i,L}(x) > 0.5)$
    - Coloriage de l'image $S(m) = i$

Nous pouvons également calculer pour chacun d'eux le score :

— Sélection de $i \in \{0, 1, 2\}$ (région à segmenter) et de $L \in \{AC, CE\}$ (fonction de perte).
— Création de la somme des scores S
— Pour chaque minibatch $x \in D_{validation}$:
    - Calcul de la prédiction $y_{pred} = U_{i,L}(x)$
    - Calcul du Dice score $S \leftarrow S + Dice(y_{pred}, y_{true})$

— Calcul de la moyenne $S/len(D_{validation})$

2.4 Résultats

Visuellement, les résultats de la segmentation automatique sont assez proches de la référence (Figure 1). D'une façon générale, nous avons remarqué que l'utilisation de la perte AC conduit à des contours plus propres, une topologie plus simple et des résultats globalement meilleurs. Nous nous sommes rendus compte que dans certains cas, la segmentation humaine ne semblait pas correcte : c'est le cas pour la dernière ligne de la Figure 1. Une potentielle amélioration serait de supprimer ces cas tangents afin d'obtenir un meilleur ensemble d'entraînement.

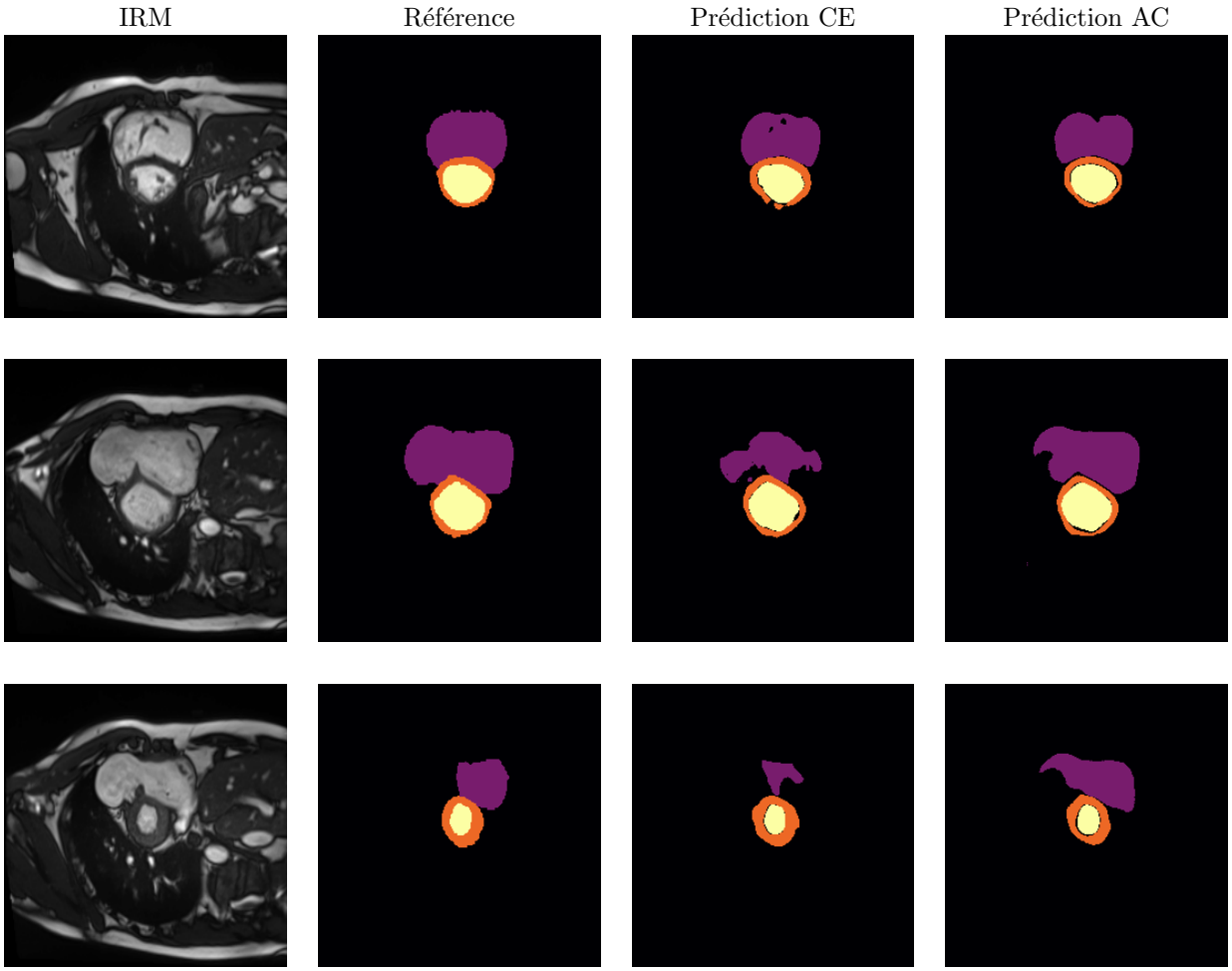


FIGURE 1 – Comparaison des segmentations

Les résultats numériques (Tableau 1) correspondent à l'analyse visuelle : l'utilisation de la loss AC permet d'obtenir un meilleure Dice Score (82.3% contre 81.8%), qui est comme prévu légèrement moins bon que celui annoncé dans l'article (96.5%). Cette différence s'explique par l'utilisation du réseau Dense U-Net, qui est 100 fois plus lent à entraîner. L'article ne donnant pas le Dice Score pour le réseau U-Net, nous ne pouvons pas directement comparer nos résultats.

2.5 Robustesse

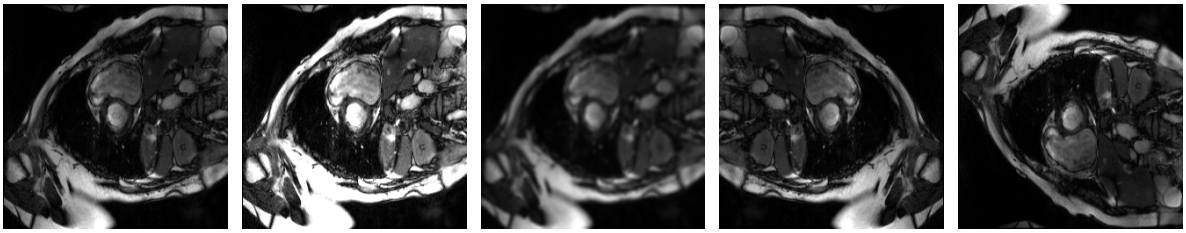
L'oeil et le cerveau humain sont extrêmement robustes et peuvent facilement généraliser. Par exemple, nous sommes capables de dire que les caractères R, R, $\mathbf{R}$, $\mathbb{R}$ correspondent tous à la même lettre de l'alphabet bien qu'ils n'aient pas la même représentation graphique. Dans le cadre de la

Réseau	Ventricule Gauche	Ventricule Droit	Myocarde	Moyenne
U-Net + CE (notre implémentation)	0.732	0.826	0.896	0.818
U-Net + AC (notre implémentation)	0.789	0.797	0.883	0.823

TABLE 1 – Dice Score obtenus dans le cadre du projet

segmentation, un médecin humain capable d'identifier les zones du coeur sur la Figure 2a sera capable d'identifier ces zones sur une image plus contrastée (Figure 2b), floutée (Figure 2c) ou retournée (Figures 2e et 2d).

Il n'en est pas de même pour les algorithmes de visions, et particulièrement pour notre réseau U-Net. Nous avons voulu estimer la robustesse de l'algorithme aux perturbations qui peuvent avoir lieu à cause de divers facteurs (paramètres d'acquisition différents, résolution plus faible..).



(a) Référence (b) Contraste (c) Flou (d) Miroir horizontal (e) Miroir vertical

FIGURE 2 – Transformations du dataset

Nous avons évalué nos trois réseaux U-Net entrainés avec la perte AC de l'article sur des images ayant subi ces perturbation. Les résultats sont réunis sur le Tableau 2. Ils nous permettent de comprendre que les images du dataset sont standardisées, car une modification du contraste ou de l'orientation perturbe significativement la segmentation. Il est intéressant de noter qu'un retournement horizontal a moins d'effet qu'un retournement vertical. Nous pensons que cela est lié à la structure du coeur, quasiment symétrique par rapport à l'axe vertical. L'ajout de flou, en revanche, n'a que peu d'effet, car il ne modifie que très peu l'histogramme des valeurs et le gradient moyen local.

Perturbation	Dice Score moyen
Aucune (référence)	0.823
Flou	0.801
Contraste	0.620
Retournement horizontal	0.463
Retournement vertical	0.129

TABLE 2 – Dice Score obtenu avec différentes perturbations

3 Conclusion

Dans ce projet, nous avons implémenté l'article étudié [CWV+19] en considérant le plus simple des deux réseaux cités, appelé U-Net. Nous avons obtenu des segmentations visuellement cohérentes, avec un Dice Score relativement proche de celui annoncé. Nous avons vu que des perturbations des images IRM en amont pouvaient conduire à de mauvaises segmentations.

Plusieurs extensions pourraient prolonger notre travail. L'augmentation automatique des données de départ (en ajoutant artificiellement des perturbations) pourrait former des réseaux plus robustes et précis. Il serait également intéressant d'utiliser ces segmentations comme input d'algorithmes déterministes (contour actif avec force de ballon par exemple) pour raffiner encore le résultat.

Références

[BBC07] Fethallah Benmansour, Stéphane Bonneau, and Laurent D Cohen. Finding a closed boundary by growing minimal paths from a single point on 2d or 3d images. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[CK97] Laurent D Cohen and Ron Kimmel. Global minimum for active contour models : A minimal path approach. *International journal of computer vision*, 24(1) :57–78, 1997.

[CMC16] Da Chen, Jean-Marie Mirebeau, and Laurent D Cohen. Finsler geodesics evolution model for region based active contours. In *Proc. BMVC*, volume 2, page 8, 2016.

[CWV+19] Xu Chen, Bryan M Williams, Srinivasa R Vallabhaneni, Gabriela Czanner, Rachel Williams, and Yalin Zheng. Learning active contour models for medical image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11640, 2019.

[LYC06] Hua Li, Anthony Yezzi, and Laurent D Cohen. 3d brain segmentation using dual-front active contours with optional user interaction. *International Journal of Biomedical Imaging*, 2006, 2006.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.