

Thomas Dargent,

SOUS LA TUTELLE DE Caroline Petitjean

WEAKLY SUPERVISED SEMANTIC IMAGE SEGMENTATION

SÉGMENTATION SÉMANTIQUE FAIBLEMENT SUPERVISÉE D'IMAGE

UNIVERSITÉ DE ROUEN NORMANDIE

Table des matières

1	<i>Introduction</i>	5
2	<i>Réalisation</i>	8
2.1	Reproduction et étude des travaux de Kervadec et al.	8
2.1.1	Prostate dataset : promise12	10
2.2	Entraînement sur ACDC.	11
2.3	Entraînement sur SegTHOR.	11
2.3.1	Pertinence de l'entropie croisée, essai de la Dice loss	13
2.3.2	CoordConv	14
3	<i>Conclusion</i>	16
	<i>Bibliographie</i>	17
	<i>Annexe</i>	19

Merci à C. Petitjean de m'avoir tutoré sur ce projet. Merci également à R. El Jurdi pour son aide.

Merci à Manon pour ses relectures.

Ce rapport a été conçu en utilisant Tufte- \LaTeX . Les figures sont issues de Matplotlib et Figma.

1

Introduction

Les images numériques que nous voyons et traitons chaque jour sont justes des matrices d'intensités de couleurs. Ce qui signifie que ce qui nous apparaît comme un animal, un objet ou une personne n'est, pour l'ordinateur qui l'affiche, qu'un tableau de nombres. Pourtant si l'on veut traiter ces images automatiquement, il faudra bien que la machine puisse en extraire plus que des valeurs de pixels individuels. "Le but de la ségmentation est de partitionner l'image en régions pour simplifier et/ou changer la représentation en quelque chose de plus simple à analyser"¹. Dans le cas d'images de texte, cela peut être de repérer les zones correspondant à des lignes, des mots ou des lettres. Pour une image naturelle on pourra chercher à isoler une personne du fond, séparer différents types d'objets (Figure 1.1)... C'est un besoin naturel, puisque c'est le rôle de notre système visuel. Pour ces derniers cas, on parle de segmentation sémantique : on cherche à tracer des zones selon le sens qu'elles contiennent.

1. Rebeca González. Image processing in python. URL <https://www.datacamp.com/courses/image-processing-in-python>

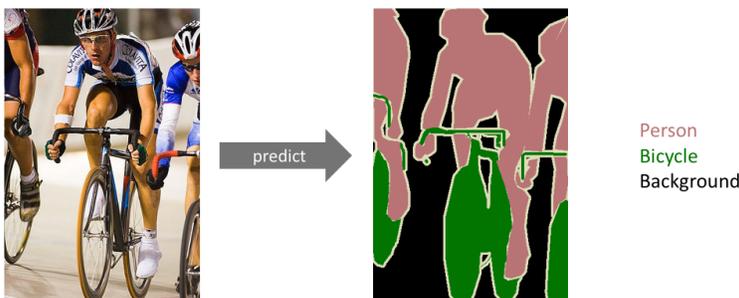


Figure 1.1: Segmentation d'image naturelle (Everingham et al., 2010)

Pour segmenter un texte tapuscrit scanné, la tâche n'est pas trop complexe. Un simple découpage en rectangles tous les x pixels peut suffire pour les cas simples. Pour un texte manuscrit c'est déjà bien plus compliqué : les lignes ne sont pas droites, les pattes peuvent se mélanger. C'est encore pire pour la majorité des images qui ne suivent pas nécessairement de règles de mise en forme. C'est pourquoi, dans beaucoup de cas, on va se tourner vers l'apprentissage

de représentation pour segmenter automatiquement une image. L'idée est de fournir à l'ordinateur suffisamment d'images d'un même type pour qu'il puisse en extraire des caractéristiques qui font sens. En comparant les images qu'on lui donne et le résultat attendu, la machine peut déduire les transformations nécessaires pour obtenir des résultats s'y approchant. Dans notre cas, on utilisera des réseaux d'apprentissage profond convolutifs. Ces réseaux sont très efficaces et sont utilisés fréquemment pour segmenter des images.

Le problème de l'apprentissage profond est qu'il nécessite un nombre énorme de données pour pouvoir généraliser. Pour certains problèmes, obtenir une telle quantité d'images est impossible. Par exemple, les images médicales nécessitent des accords entre les laboratoires de recherches en intelligence artificielle et les hôpitaux ce qui peut être complexe. De plus, et c'est peut être le problème majeur, pour apprendre la machine a besoin d'avoir une vérité terrain comme point de comparaison.² Pour obtenir ces vérités terrains, il faut qu'un médecin prenne le temps de tracer la segmentation correcte pour chaque image. On parle de milliers d'images, ce savoir est donc coûteux.

2. La vérité terrain correspond au résultat que l'on aimerait avoir idéalement en sortie du réseau.

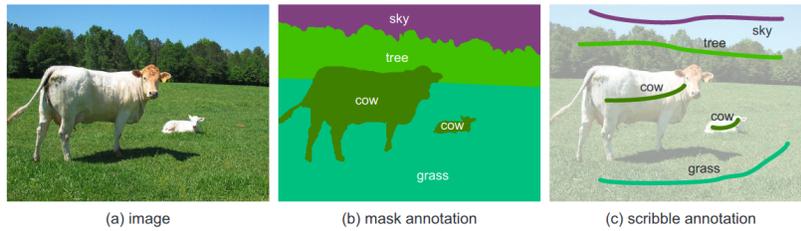


Figure 1.2: Exemple de segmentation sémantique faiblement supervisée basée sur un trait, Figure venant de (Lin et al., 2016)

Afin de réduire ce travail pénible, on peut essayer de définir plus faiblement les vérités terrains. Au lieu de définir pour chaque pixel la zone à laquelle il appartient, on définit par exemple seulement un sous ensemble de ces pixels.^{1.2} On peut également tracer un cadre entourant de manière plus ou moins large la zone, ou, dans les cas les plus extrêmes, juste donner le nom des choses présentes dans l'image. La tâche devient clairement plus aisée pour l'expert devant fournir la vérité terrain. En revanche le réseau a moins d'informations sur la distribution de sortie, et donne donc en général des résultats moins pertinents.

Dans ce rapport, on explorera le travail de (Kervadec et al., 2018), qui propose d'ajouter une contrainte sur la taille de la zone. En ajoutant ainsi une information au réseau, ils ont pu obtenir de très bons résultats sur des tâches de segmentation sémantique faiblement supervisées. On tentera de reproduire leurs résultats, et de les transposer à un autre jeu de données. Enfin la question que l'on se posera est « Peut on s'approcher encore plus d'un score entièrement supervisé? ». On explorera différentes idées et l'on conclura sur leurs performances.

On évaluera la réussite de nos segmentations à partir de l'indice de Sørensen-Dice. Il mesure la superposition entre la zone prédite et la zone réelle : à 1, les zones sont confondues, à 0 elles sont complètement séparées. Il se formule

ainsi :

$$(1.1) \quad \text{DSC} = \frac{2|X \cap Y|}{|X| + |Y|}$$

Comme on traite des valeurs binaires on peut le voir comme :

$$(1.2) \quad \text{DSC} = \frac{2 \sum_{i \in \Omega} p_i g_i}{\sum p_i + \sum g_i}$$

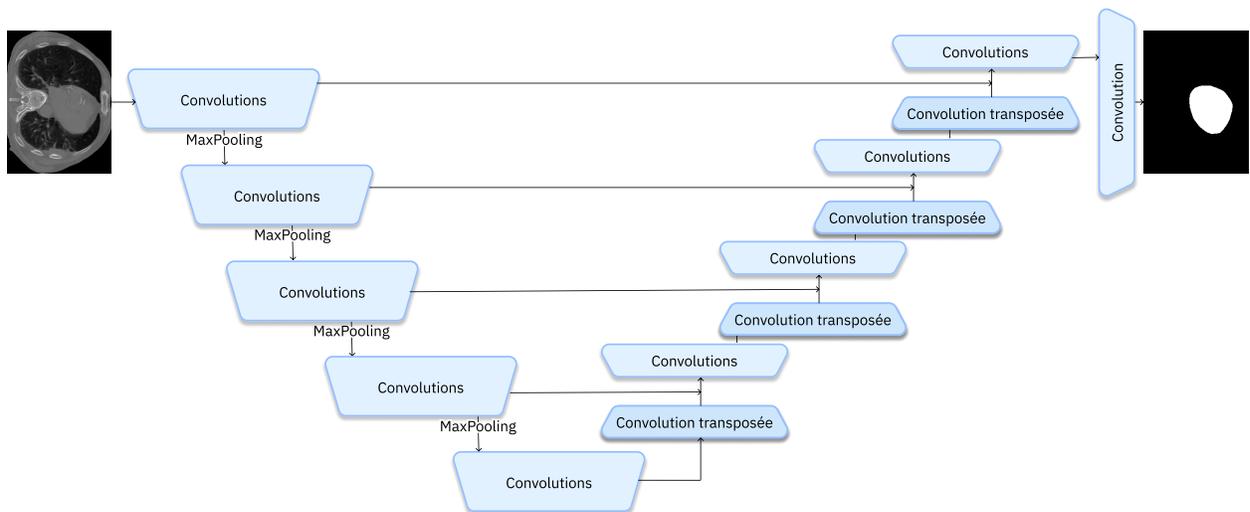
$$(1.3) \quad = \frac{2 \sum_{i \in \Omega_{GT}} p_i}{\sum p_i + \sum g_i}$$

Où les p_i sont les pixels prédits, les g_i sont les pixels de la vérité terrain, Ω est l'ensemble des pixels et Ω_{GT} l'ensemble des pixels où la vérité terrain est égale 1 (donc où il y a une zone).

2

Réalisation

2.1 Reproduction et étude des travaux de Kervadec et al.



Dans un premier temps on a tenté de reproduire les résultats de (Kervadec et al., 2018)¹ et de comprendre leurs travaux. Ils ont entraîné un modèle d'apprentissage profond sur différents jeux de données médicaux afin de tester la validité de leurs résultats. Le réseau entraîné est basé sur une architecture U-net (Figure 2.1). Un U-net est un réseau souvent utilisé pour de la segmentation. Comme son nom l'indique, il prend la forme d'un "U" :

1. La première partie des traitements est dite "contractive". Une succession de convolutions et de poolings va réduire la taille des données en extrayant leurs caractéristiques. Après chaque convolution, les données sont plus abstraites, la représentation est plus haut niveau. En somme, cette phase va extraire dans le contexte de chaque pixel le sens que l'on souhaite obtenir.
2. La seconde partie des traitements est "expansive". L'idée est de continuer à

Figure 2.1: Schéma du réseau utilisé. Les blocs de convolutions utilisés sont en réalité plus complexes, puisqu'il s'agit d'enchaînements de multiples convolutions avec des BatchNorm.
1. Hoel Kervadec, Jose Dolz, Meng Tang, Eric Granger, Yuri Boykov, and Ismail Ben Ayed. Constrained-cnn losses for weakly supervised segmentation. *CoRR*, abs/1805.04628, 2018. URL <http://arxiv.org/abs/1805.04628>

effectuer des convolutions sur les caractéristiques extraites, mais en faisant des convolutions transposées² pour reconstruire l'espace de l'image originale. Enfin, entre chaque convolution transposée, on va faire la moyenne entre la sortie de ces dernières avec celles des convolutions équivalentes de la première phase et faire une convolution sur le résultat.

Si ces réseaux sont très efficaces pour de la segmentation, un entraînement classique en semi-supervisé serait peu probant. Les auteurs ont trouvé une technique innovante pour obtenir de meilleurs résultats en faiblement supervisé. L'idée est assez logique : Lorsque l'on traite des images médicales, celles-ci sont normées, elles ont des proportions standardisées. De plus, comme les organes humains varient peu en taille, on peut s'attendre à des segmentations de tailles relativement équivalentes. Les auteurs ont donc choisi de contraindre la taille de la zone segmentée à une intervalle observée empiriquement. Ce faisant ils ajoutent une connaissance préalable (*prior knowledge*) de la distribution de sortie dans le réseau, rendant l'entraînement plus simple. Pour ajouter une contrainte aux résultats obtenus ils ont modifié la fonction de coût.

Observons la fonction choisie. On pose S la sortie Softmax du réseau (qui peut être vue comme une prédiction de la probabilité d'appartenance à la zone voulue) :

$$(2.1) \quad L = \mathcal{H}(S) + \lambda \mathcal{C}(V_s)$$

La première partie de l'équation est assez classique, il s'agit du calcul de l'entropie croisée entre les pixels correctement supervisés et la sortie du réseau. Comme on cherche à minimiser L , dans le cas idéal $\mathcal{H}(S) = 0$, or cela arrive quand les deux distributions sont égales. Cette première partie s'assure donc que l'entraînement amène le réseau à classifier correctement les pixels supervisés.

La seconde est la partie proposée par (Kervadec et al., 2018). Cette régularisation va contraindre la taille de la sortie. On a $V_s = \sum_p S_p$, donc V_s peut être interprété comme l'aire de la zone segmentée par le réseau. On définit les contraintes de tailles déterminées empiriquement telle que $a \leq \sum_{p \in \Omega_s} p \leq b$, où Ω_s est l'ensemble des pixels des zones à segmenter.³

$$\mathcal{C}(V_s) = \begin{cases} (V_s - a)^2, & \text{si } V_s < a \\ (V_s - b)^2, & \text{si } V_s > b \\ 0 & \text{sinon} \end{cases}$$

On constate que plus V_s sort de l'intervalle choisi, plus $\mathcal{C}(V_s)$ est grand. Comme la rétropropagation vise à réduire le coût, on voit bien que cela signifie conserver une zone dans l'intervalle bornée par a et b .

En pratique les labels affaiblis sont générés par le script de préparation de données à partir de la vérité terrain correspondante. On a sélectionné trois types d'entraînements pour chaque jeu de données 2.2 :

2. Une convolution transposée est une convolution classique mais en ajoutant du padding entre les pixels d'entrée. Cela permet d'avoir une convolution dont la sortie est de taille similaire ou plus grande que l'entrée

3. On pourra les définir comme les tailles minimales et maximales des zones, sur un ensemble de vérités terrains entièrement annotées.

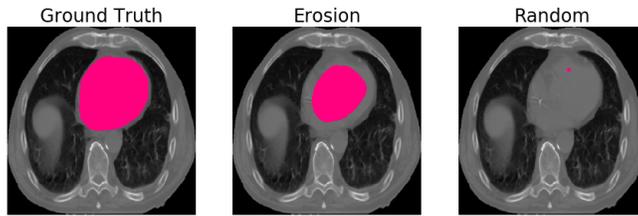


Figure 2.2: Les différents types de labels utilisés.

1. **Affaiblissement par érosion** : On réduit la taille de la zone en lui faisant subir une érosion entre 0 et 10 fois. Sachant qu'au final on prendra la zone la plus réduite. On érode avec une structure carrée de 3x3 pixels. On a donc une réduction maximale d'environ 30 pixels sur le rayon d'un cercle.
2. **Affaiblissement aléatoire** : On prend comme label un cercle de diamètre déterminé (on a pris 4 pixels) que l'on place au hasard dans la zone de la vérité terrain.
3. **entièrement supervisé** : On prend comme label la vérité terrain. Dans ce cas on utilisera pas la contrainte SizeLoss, afin de suivre les scénarios choisis par (Kervadec et al., 2018)

2.1.1 Prostate dataset : promise12

Le code utilisé par les auteurs est disponible librement en ligne sur GitHub.⁴ On a dans un premier temps tenté de lancer les mêmes entraînements qu'eux. On a sélectionné le jeu de données PROMISE12⁵. Il s'agit de scans IRM de prostates rendus publiques lors du challenge "MICCAI 2012". Les images viennent de différentes machines, et ont été obtenues selon des protocoles différents. On a 50 patients avec une maladie bénigne ou un cancer de la prostate. Ces données sont augmentées :

1. Redimensionnées en 256×256 pour des raisons de conservation de mémoire,
2. Pour chaque image, on crée 4 images augmentées,
3. Elles sont inversées verticalement et/ou horizontalement, et/ou tournées de 0 à 45°.

Cette phase, de prime abord simple (il s'agit de lancer un entraînement "clé en mains"), a pris plus de temps que prévu. Il a fallu comprendre les conditions d'entraînements et commencer à s'appropriier le code. Au final les scores semblent être moitié moins élevés que ceux indiqués dans l'article. On peut expliquer ça par une divergence entre nos images et celles utilisées par les auteurs du code, car elles n'étaient pas fournies dans le répertoire github. Il est probable que nous les ayons téléchargé sur le même site (celui du challenge) donc

4. https://github.com/LIVIAETS/SizeLoss_WSS

5. Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, Robin Strand, Filip Malmberg, Yangming Ou, Christos Davatzikos, Matthias Kirschner, Florian Jung, Jing Yuan, Wu Qiu, Qinquan Gao, Philip "Eddie" Edwards, Bianca Maan, Ferdinand van der Heijden, Soumya Ghose, Jhimli Mitra, Jason Dowling, Dean Barratt, Henkjan Huisman, and Anant Madabhushi. Evaluation of prostate segmentation algorithms for mri: The promise12 challenge. *Medical Image Analysis*, 18(2): 359 – 373, 2014. ISSN 1361-8415. doi: <https://doi.org/10.1016/j.media.2013.12.002>. URL <http://www.sciencedirect.com/science/article/pii/S1361841513001734>

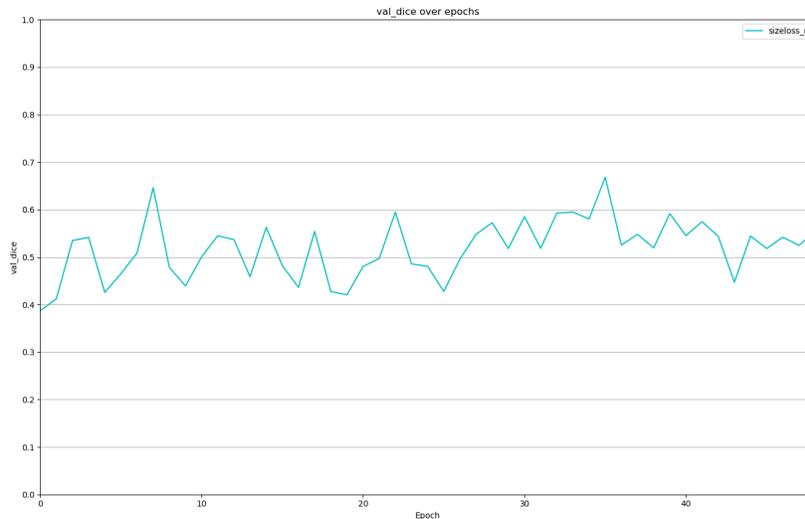


Figure 2.3: Evolution de la fonction de coût en validation sur Promise, avec un entraînement sur des labels aléatoires

la chance que ce soit la cause est faible. On a aussi effectué peu d'entraînements (un pour l'érosion, un pour l'aléatoire) car ces derniers sont très longs sur ce jeu de données. Les deux entraînements fait ont tous les deux été coupés avant leurs fin (qui était définie à 200 epochs), mais ne montraient pas de signes d'améliorations (Figure 2.3). Au final, comprendre la cause de cet echec demanderait une investigation plus poussée. Toutefois, cela indique que nous sommes capables de lancer un entraînement selon le scénario *Partial CE + Tags + Common Size*.

2.2 Entraînement sur ACDC

Afin de s'assurer que nos résultats ont du sens, il nous faut quand même vérifier si l'on arrive à faire concorder des scores avec ceux obtenus par (Kervadec et al., 2018). Dans le cas contraire, on pourra mettre la réussite de tous les résultats ultérieurs sur une facilité plus grande des jeux de données utilisés. On a donc également essayé des scénarios d'entraînements sur le jeu Automated Cardiac Diagnosis Challenge. Il s'agit d'image de scans issus d'IRM. Ces scans montrent 4 différentes pathologies, ainsi qu'un groupe sain.

Après un entraînement de 163 epochs (sur 200 prévues) en érosion on trouve 0.6944 de DICE en validation. Ce score est plutôt proche de celui obtenu par les auteurs, on a donc notre réponse. ACDC et promise12 contiennent plus d'images et prennent beaucoup plus de temps à converger. Les mauvais résultats semblent surtout tenir des interruptions trop rapides des entraînements⁶. On trouve également 0.8598 de Dice en validation après un entraînement complet.

2.3 Entraînement sur SegTHOR

6. Ces interruptions sont surtout dues au type de machine sur laquelle elles surviennent. On utilise une machine virtuelle préemptive de Google, elle s'interrompt donc dès qu'il y a une surcharge serveur.

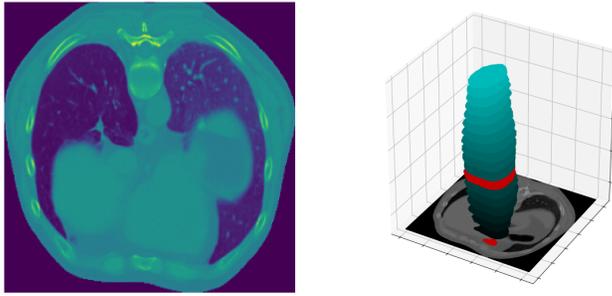


Figure 2.4: Représentation des différentes tranches d'un patient. On voit à droite les différentes vérités terrains du même patient se superposer les unes sur les autres. En rouge sont marquées la plus petite et la plus large (respectivement la plus basse et une de celles vers le milieu). Un tiers des scans seulement sont affichés, afin de gagner en clarté. À gauche se trouve une des images qui sera à segmenter, se trouvant au milieu des scans du patient représenté ici.

On a donc tenté des entraînements sur de nouveaux jeux de données. On a pu utiliser une sous partie du dataset SegTHOR⁷. À la base SegTHOR est une compétition dont le but est de segmenter automatiquement quatre organes à risque (cœur, aorte, trachée, oesophage) sur des images de tomodensitométrie. L'idée est que lors de traitements par radiothérapie, certains organes doivent à tout prix être évité car il risquerait de mal supporter le traitement. Le problème est que c'est le praticien qui doit tracer les zones à éviter. Pendant ce temps le patient et ses organes peuvent bouger, et ce processus n'est donc pas précis⁸. Une délimitation des zones par ordinateur serait plus rapide et donc potentiellement plus pertinente.

Pour notre application on a les images de 29 patients dont le cœur seulement est annoté, la technique utilisée ne permettant pas la multisegmentation. Cela représente un total de 391 couples image/vérité terrain. On a découpé ces données en 219 images d'entraînements, et 153 de validations, après augmentation on a donc 1248 images : 1095 d'entraînements et 153 de validations. Ces images sont regroupées par patient, et anonymisées avec un nom de code. Elles sont triées par leur hauteur dans le torse du patient (voir Figure 2.4). En moyenne on a 13.48 tranche par patients.

Pour pouvoir les utiliser, on a du redimensionner les images à une taille fixe, les normaliser par patient, et globalement s'assurer qu'elles s'adaptent bien aux algorithmes fournis. Empiriquement on a trouvé les bornes suivantes : entre 217px et 7713px. Comme précédemment, on a lancé des entraînements avec des labels érodés, aléatoires et entièrement supervisés. Les résultats ont été satisfaisants (Table 2.1). L'entraînement entièrement supervisé sert en quelque sorte de borne haute aux entraînements faiblement supervisés. En effet, il montre ce dont est capable un tel réseau sur ces données quand on lui donne la totalité des informations à notre disposition. On ne s'attend pas à ce qu'avec des labels lacunaires le réseau fasse mieux⁹. On peut alors imaginer le rapport $\frac{\text{Weak}}{\text{Full}}$ qui nous donne une idée d'à quel point le savoir a priori permet de se rapprocher d'un entraînement entièrement supervisé. Ici on est à 0.9231 pour

7. Roger Trullo, Caroline Petitjean, Bernard Dubray, and Su Ruan. Multiorgan segmentation using distance-aware adversarial networks. *Journal of Medical Imaging*, 6(1):1 – 11, 2019. doi: 10.1117/1.JMI.6.1.014001. URL <https://doi.org/10.1117/1.JMI.6.1.014001>

8. C. F. Njeh. Tumor delineation: The weakest link in the search for accuracy in radiotherapy. *Journal of medical physics*, 33(4):136–140, Oct 2008. ISSN 1998-3913. doi: 10.4103/0971-6203.44472. URL <https://www.ncbi.nlm.nih.gov/pubmed/19893706>. 19893706[pmid]

9. Il faut également se rappeler que les tracés experts peuvent être différents. H. Kervadec cite ici un Dice entre expert de 90% : https://github.com/LIVIAETS/SizeLoss_WSS/issues/2#issuecomment-537393797. Le chiffre est une grosse approximation et varie selon la tâche mais il peut également servir de borne haute.

un entraînement avec labels érodés, et 0.9246 avec labels aléatoires. Dans l'idéal on comparerait ce score avec des scores sur des entraînements sans contraintes. Ces entraînements par ablation nous assureraient que ces résultats puissent être attribués aux contraintes Sixeloss, mais on a pas pu les réaliser dans le temps imparti. Des résultats préliminaires sur un tel entraînement (à 73 epochs seulement) nous donne un score maximal en validation de 0.5391. Cela nous fait un rapport de 0.6066, ce qui semble indiquer que l'ajout de contraintes est en effet déterminant dans ces scores. En attendant, on peut être satisfait que ces rapports soit aussi hauts.

entièrement Supervisé	Label Erodés	Label aléatoire
0.8887	0.8204	0.8217

Table 2.1: Meilleur DICE score en validation à la meilleure des epochs pour SegTHOR

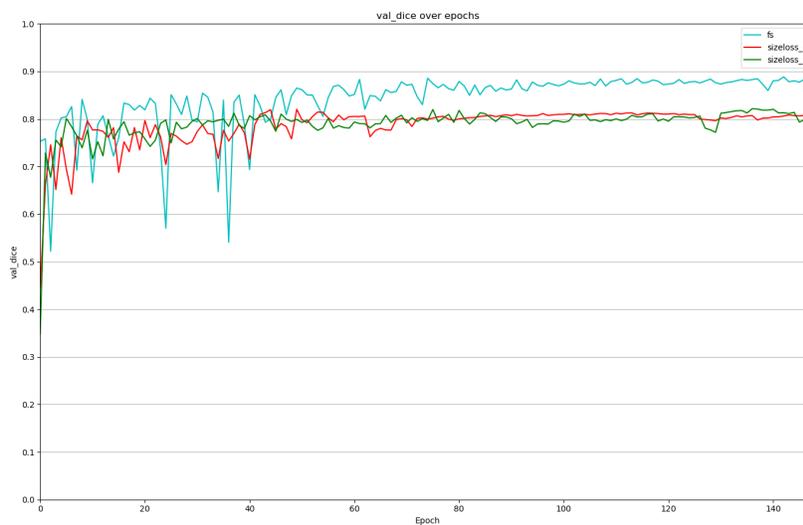


Figure 2.5: Evolution du score de validation sur SegTHOR avec affaiblissement des labels érodés (bleu), aléatoires (vert) et sans affaiblissement (rouge)

2.3.1 Pertinence de l'entropie croisée, essai de la Dice loss

La métrique utilisée pour juger de la réussite de l'apprentissage est l'indice de Sørensen-Dice (Voir l'équation 1.2). Dans d'autres travaux, une variante de ce coefficient est également utilisée comme fonction de coût. On notera DSC la fonction qui retourne le coefficient de Dice. Comme $\max(\text{DSC}) = 1$, et que l'on cherche à le maximiser, alors on veut minimiser :

$$(2.2) \quad \text{Dice Loss}(p) = 1 - \text{DSC}(p) = 1 - \frac{2 \sum_{i \in \Omega_{GT}} p_i}{\sum p_i + \sum g_i}$$

On a d'abord tenté une simple substitution entre l'entropie croisée et cette fonction de coût. L'équation 2.1 devient alors :

$$(2.3) \quad L = \text{Dice Loss}(s) + \lambda \mathcal{C}(V_s)$$

Seulement les résultats furent peu satisfaisants (Equivalent à la moitié des scores précédents). On pourrait également utiliser un mélange des deux loss de manière pondérée (Equation 2.4), mais le temps a manqué pour en mesurer l'impact.

$$(2.4) \quad L = \mathcal{H}(S) + \gamma \text{Dice Loss}(s) + \lambda \mathcal{C}(V_s)$$

2.3.2 CoordConv

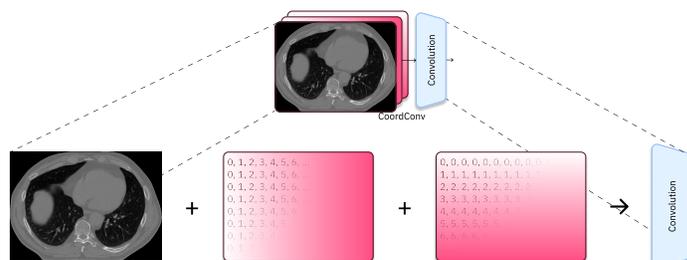


Figure 2.6: Fonctionnement d'un bloc CoordConv. On concatene avec l'image, des couches representant l'évolution en x ou y, puis on effectue une convolution sur le tout.

La segmentation vise à trouver dans un espace donné, des sous espaces délimités. (Liu et al., 2018)¹⁰ montrent que les réseaux convolutifs ne sont pas aussi bons que ce que l'on pourrait penser sur des tâches comprenant de la localisation. En expérimentant sur un problème apparemment simple (passer d'une coordonnée à une représentation cartésienne), ils ont obtenu des résultats médiocres avec un réseau convolutif classique. Notamment lorsque le réseau est entraîné sur des zones restreintes de l'image, il est incapable de généraliser aux autres zones. Ils proposent alors une solution simple et peu coûteuse : Ajouter deux canaux à chaque image avant la convolution. Ces canaux supplémentaires sont la matrice des coordonnées en x et y de la matrice reçue. Étonnamment ce simple ajout semble leur avoir donné de très bons résultats pour les problèmes étudiés.

Encouragés par ces résultats, on a tenté d'implémenter ce type de convolution dans notre réseau (Figure 2.8). Comme eux, on s'est contenté de modifier la première convolution (Le code est disponible en annexe). Les résultats ne semblent pas significativement meilleurs, comme on peut le voir dans le tableau 2.2.

	entièrement Supervisé	Label Erodés	Label aléatoire
Avec CoordConv	0.8937	0.8104	0.8275
Sans CoordConv	0.8887	0.8204	0.8217

10. Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. *arXiv e-prints*, art. arXiv:1807.03247, Jul 2018

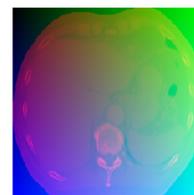


Figure 2.7: L'ajout de ces couches peut être vu comme l'ajout des canaux verts et bleus sur une image en nuance de gris. On voit ici : en rouge l'image originale, en bleu la couche "y" et en vert la couche "x"

Table 2.2: Meilleur DICE score en validation à la meilleure des epochs sur SegTHOR avec et sans Coord Conv

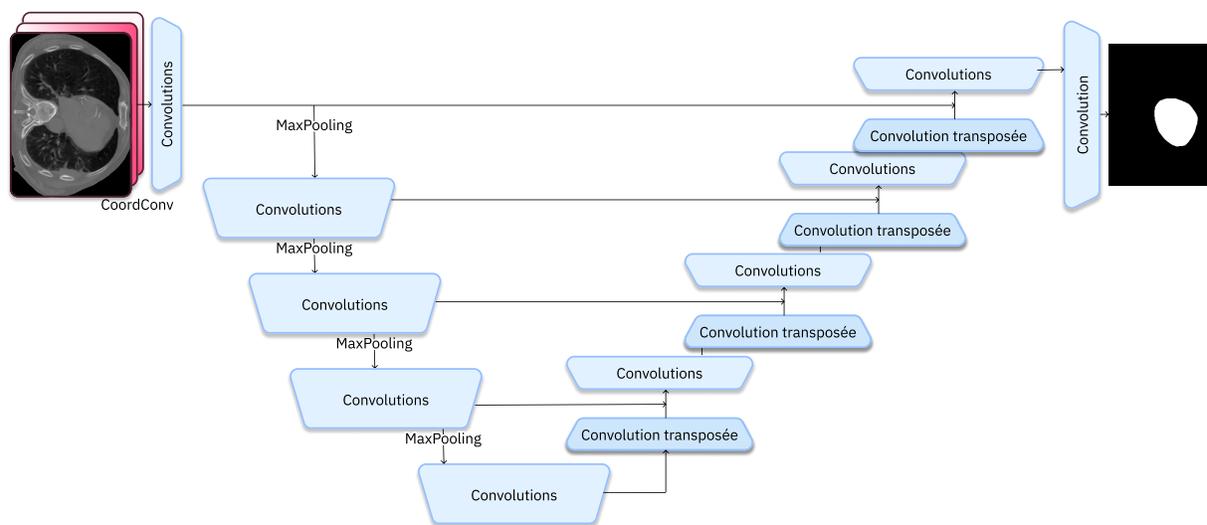


Figure 2.8: Schéma de notre réseau après l'ajout d'un bloc CoordConv.

3

Conclusion

Au final, bien que nous ayons eu des résultats mitigés sur les jeux de données utilisés dans l'article original, on a pu adapter la méthode de (Kervadec et al., 2018) au dataset SegTHOR. Les résultats semblent bons, mais il est fort probable que ce sous échantillon du jeu de données soit trop simple. En effet les résultats aléatoires sont équivalents aux résultats avec érosion, alors que la taille de la zone est très différente! Cela semble indiquer que dans ce cas, la loss contribue de manière majoritaire à l'entraînement. Pour le vérifier il faudra terminer un entraînement faiblement supervisé sans cette fonction de coût, mais un entraînement préliminaire semble confirmer cette hypothèse.

L'ajout d'une DiceLoss n'a pas été probant, mais demande plus d'exploration avant d'être définitivement écarté. Il faudra essayer une fonction de coût s'exprimant comme une somme pondérée de l'entropie croisée, de la DiceLoss et de la contrainte SizeLoss comme indiqué dans l'équation 2.4. On pourra, par une GridSearch ou une recherche aléatoire, trouver les valeurs idéales de ces paramètres et conclure.

La CoordConv n'a pas amélioré significativement les résultats obtenus sur SegTHOR. En revanche on pourrait essayer de la placer à différents emplacements dans le réseau, et de modifier les paramètres des convolutions. De plus il faudra faire des essais sur d'autres datasets (promise12, ACDC...) pour vérifier si les conclusions dépendent uniquement de SegTHOR ou non.

Enfin le point noir majeur de tous nos résultats est l'absence de score sur un ensemble de test. On a jamais réussi à faire marcher l'option présente dans le code utilisé. Cela diminue le poids des résultats présentés. Il faudra absolument résoudre ce problème pour pouvoir conclure avec plus de certitudes.

Dans un cadre plus général, SegTHOR étant à la base conçu pour la segmentation multi-classe (détection de plusieurs organes par image), Il serait intéressant de traiter ce problème, en y appliquant les connaissances apprises ici.

Bibliographie

- M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2) :303–338, June 2010.
- Rebeca González. Image processing in python. URL <https://www.datacamp.com/courses/image-processing-in-python>.
- Hoel Kervadec, Jose Dolz, Meng Tang, Eric Granger, Yuri Boykov, and Ismail Ben Ayed. Constrained-cnn losses for weakly supervised segmentation. *CoRR*, abs/1805.04628, 2018. URL <http://arxiv.org/abs/1805.04628>.
- Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup : Scribble-supervised convolutional networks for semantic segmentation. 04 2016.
- Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerckstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, Robin Strand, Filip Malmberg, Yangming Ou, Christos Davatzikos, Matthias Kirschner, Florian Jung, Jing Yuan, Wu Qiu, Qinquan Gao, Philip “Eddie” Edwards, Bianca Maan, Ferdinand van der Heijden, Soumya Ghose, Jhimli Mitra, Jason Dowling, Dean Barratt, Henkjan Huisman, and Anant Madabhushi. Evaluation of prostate segmentation algorithms for mri : The promise12 challenge. *Medical Image Analysis*, 18(2) :359 – 373, 2014. ISSN 1361-8415. doi: <https://doi.org/10.1016/j.media.2013.12.002>. URL <http://www.sciencedirect.com/science/article/pii/S1361841513001734>.
- Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. *arXiv e-prints*, art. arXiv :1807.03247, Jul 2018.
- C. F. Njeh. Tumor delineation : The weakest link in the search for accuracy in radiotherapy. *Journal of medical physics*, 33(4) :136–140, Oct 2008. ISSN 1998-3913. doi: 10.4103/0971-6203.44472. URL <https://www.ncbi.nlm.nih.gov/pubmed/19893706>. 19893706[pmid].
- Roger Trullo, Caroline Petitjean, Bernard Dubray, and Su Ruan. Multiorgan segmentation using distance-aware adversarial networks. *Journal of Medical*

Imaging, 6(1):1 – 11, 2019. doi: 10.1117/1.JMI.6.1.014001. URL <https://doi.org/10.1117/1.JMI.6.1.014001>.

Annexe

Code utilisé pour intégrer CoordConv

```
1 class Coord_Block(nn.Module):
2     def __init__(self, centered=True):
3         self.centered = centered
4         use_cuda = torch.cuda.is_available()
5         self.device = torch.device('cuda:0' if use_cuda else 'cpu')
6         super().__init__()
7
8     def forward(self, input):
9         """
10        Concatenate additional channels to the input. Those channels are simply
11        range increasing on each of the 2D dimensions.
12        """
13        batch_size, _nb_channel, x_dim, y_dim = input.shape
14
15        xx_range = torch.arange(y_dim, dtype=torch.int32).to(self.device).repeat(x_dim, 1)
16        assert xx_range.shape == (x_dim, y_dim)
17        xx_range = xx_range.repeat(batch_size, 1, 1)
18        assert xx_range.shape == (batch_size, x_dim, y_dim)
19        assert xx_range[0, 0, 0] == xx_range[0, 1, 0]
20        assert xx_range[0, 0, 0] == xx_range[0, 0, 1] - 1 # range is on idx 3
21        xx_range = xx_range.unsqueeze(1).type(dtype=torch.float32)
22        xx_range = xx_range / (x_dim - 1)
23
24        yy_range = torch.arange(x_dim, dtype=torch.int32).to(self.device).repeat(y_dim, 1)
25        assert yy_range.shape == (y_dim, x_dim)
26        yy_range = yy_range.repeat(batch_size, 1, 1)
27        assert yy_range.shape == (batch_size, y_dim, x_dim)
28        assert yy_range[0, 0, 0] == yy_range[0, 1, 0]
29        assert yy_range[0, 0, 0] == yy_range[0, 0, 1] - 1 # range is on idy 3
30        yy_range = yy_range.unsqueeze(1).type(dtype=torch.float32)
31        yy_range = yy_range.permute(0, 1, 3, 2)
32        yy_range = yy_range / (y_dim - 1)
33
34        if self.centered:
35            xx_range = 2*xx_range - 1
36            yy_range = 2*yy_range - 1
37
38        return torch.cat((input, xx_range, yy_range), 1) # (batch_size, channels, x, y)
39
40 def coord_conv_block(in_dim, out_dim, act_fn, kernel_size=3, stride=1, padding=1, dilation=1):
41     model = nn.Sequential(
42         Coord_Block(),
43         nn.Conv2d(in_dim+2, out_dim, kernel_size=kernel_size, stride=stride, padding=padding, dilation=dilation),
```

```
44     nn.BatchNorm2d(out_dim),
45     act_fn,
46 )
47 return model
48
49
50
51 class CoordConv_residual_conv(nn.Module):
52     def __init__(self, in_dim, out_dim, act_fn):
53         super().__init__()
54         self.in_dim = in_dim
55         self.out_dim = out_dim
56         act_fn = act_fn
57
58         self.conv_1 = coord_conv_block(self.in_dim, self.out_dim, act_fn)
59         self.conv_2 = conv_block_3(self.out_dim, self.out_dim, act_fn)
60         self.conv_3 = conv_block(self.out_dim, self.out_dim, act_fn)
61
62     def forward(self, input):
63         conv_1 = self.conv_1(input)
64         conv_2 = self.conv_2(conv_1)
65         res = conv_1 + conv_2
66         conv_3 = self.conv_3(res)
67         return conv_3
```