Projet 3A: Segmentation d'images

Te Sun, Ge Jin
Tuteur: Stephanie Allassonniere

Mars. 2019

Codes :`https://github.com/sun-te/MAP512_Segmentation`

1 Sommaire

Segmentation d'images est très présente dans le domaine du traitement d'image. En imagerie médicale, le développement des méthodes dites *Machine Learning* émerge comme un challenger devant les approches algorithmiques basées sur des arguments mathématiques ou anatomiques. Dans le cardre de ce rapport, nous allons comparer les deux méthodes : *Random Walker*[1] et *AtlasNet* [2]. Et nous cherchons à donner une analyse synthétique comme une suggestion pour la segmentation.

Mots Clés : Segmentation d'images, réseaux de neurones, marche aléatoire.

2 Introduction

Segmentation d'image est un des problèmes les plus étudiés en l'analyse des imageries médicales. Le découpage d'images essaye d'identifier les zones *similaires*. On souhaite étudier dans ce projet deux méthodes proposées : 1.*Random Walker* : une méthodes de segmentation basée sur des idées de marche aléatoire déjà utilisée dans un cadre d'imagerie médicale. 2.*AtlasNet* : une approche de segmentation par apprentissage profonde et par de multiples transformations non-linéaires.

Dans le cadre de ce projet, nous chercherons, en premier lieu, à comprendre ces deux méthodes. En deuxième lieu, nous allons appliquer ou ré-implémenter ces deux approches sur les données d'entraînement générés. À la fin, une comparaison entre différentes approches sera réalisée afin de proposer des approches plus adaptées pour la segmentation d'images médicales.

Le rapport est construit en trois parties. Dans la section *Méthode de Segmentation*, nous représenterons les deux méthodes, à savoir, *Random Walker* et *AtlasNet* au niveaux théorique. Ensuit, les détailles d'implémentation serons développé dans la partie *Expérience*, où nous présenterons les données utilisés et les codes de segmentation. Nous conclurons par la comparaison des deux méthodes et par une proposition des amélioration potentielle.

3 Méthodes de Segmentation

3.1 Random Walker

Random Walker [1] est une méthodes proposées en 2006 par *Leo Grady*. Comme il s'agit d'une méthodes non-apprentissage, sa performance est indépendant du nombre de données. Cette méthodes est donc caractérisée par un fort robustesse. Pour cela, il lui suffit de spécifier quelques points à l'intérieur d'une certaine zone ainsi que quelques points à l'extérieur (Fig 1).

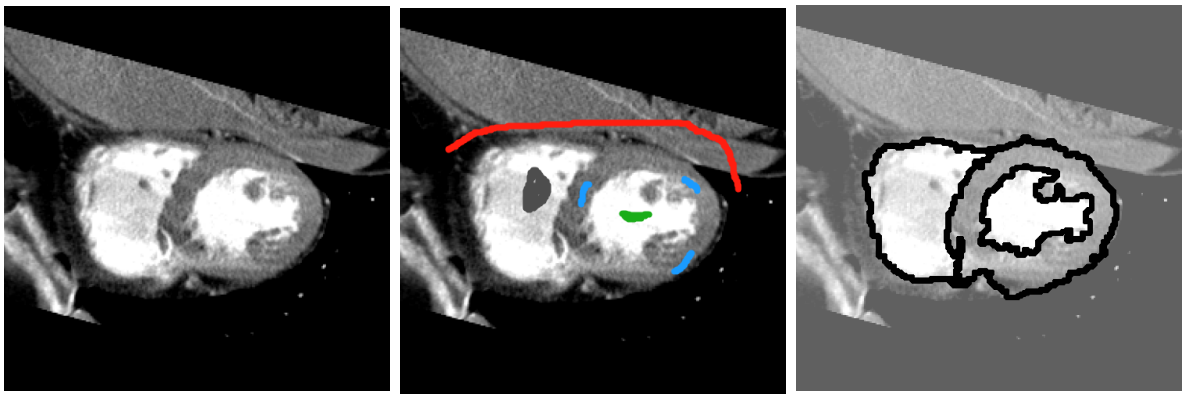


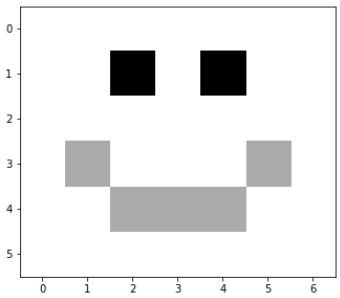
Image originale | Labels des zones | Segmentation

Figure 1 – Segmentation d'une tomographie du muscle cardiaque par *Random walker*

3.1.1 Représentation de l'algorithme

Considérons une matrice de tailles 7×7 ci dessous :

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Cette matrice représente donc une image à gauche : une image de taille 7×7 pixels et possède une **intensité** g_i à chaque pixel. On a donc une image en niveau de gris (au lieu d'une image RGB) qui n'a qu'un seul 'channel'. Grâce à cette notation, l'image sera transformé comme un graphe discret dont chaque **sommet V**(vertice) est liée par un **arête non-orienté E**(edge). En pratique, on peut définir les voisins d'un sommets de façons suivante : 1. **4-voisins modèles** : un sommet $P_{ij}(i, j$ signifie sa position dans la matrice) admet $P_{<i-1,j>}$, $P_{<i,j-1>}$, $P_{<i+1,j>}$, et $P_{<i,j+1>}$ comme voisins ; 2. **8-voisins modèles** : un sommet admet les quatre sommets d'avant et les voisins diagonaux comme voisins : $P_{<i-1,j-1>}$, $P_{<i-1,j+1>}$, $P_{<i+1,j-1>}$, et $P_{<i+1,j+1>}$. Où $< i, j >= \Big(\max(0, \min(i, W)), \max(0, \min(j, H))\Big)$. H et W présente la taille de l'image.

Un arête qui relie V_i et V_j possède un **poid** w_{ij}. On peut dès lors définir le **degré** d_i qui est la sommations du poids des arêtes issues de ce point. Dans le but de segmenter une image, on souhaite que plus la différence de l'intensité entre deux sommets est importante, moins le poids de cet arête est présent. Dans l'article [1], Grady prends le poids Gaussian en raison empirique, et qui est défini comme :

$$w_{i,j} = \exp(-\beta(g_i - g_j)^2)$$

β est un paramètre à choisir selon la performance. Un grand β va rendre l'image plus homogène comme on va réduire l'écart des poids parmi les arêtes. Il est possible d'être réglé à posteriori.
Remarque : cette définition est généralisable quand l'image est de **RGB**. Il suffit de remplacer le carré par une norme $||g_i - g_j||^2$ comme la différence d'intensité.

En passant dans un domaine continue, la différence d'intensité est une équivalence du **gradient**. Le but, c'est de trouver des zones dans lesquels l'intégral de la norme du gradient est minimum. Le problème de segmentation devient une minimisation d'un **intégral Dirichlet** :

$$D[u] = \frac{1}{2}\int_\Omega |\nabla u|^2 \mathrm{d}\omega$$

avec les conditions aux bords définies par des labels marqués manuellement.

Dans le monde discret, l'intégral Dirichlet peut être exprimé à l'aide de la matrice Laplacienne :

$$L_{i,j} = \begin{cases} d_i & \text{Si } i = j, \\ w_{i,j} & \text{Si } V_i,\ V_j \text{ sont voisins} \\ 0 & \text{autres cas} \end{cases}$$

et l'intégral Dirichelet :

$$D[X] = \frac{1}{2}X^T.L.X$$

avec $G = [x_1, x_2, \cdots, x_{H\times W}]^T$, x_i est label du sommet i. On peut facilement montrer que cette quantité est invariante par toute permutation des rangs sur X et L (en même temps). Donc si l'on note G_n les intensités des sommets non-masqués (non-labellés) et G_m, celles des points avec labels, on aura :

$$D[X] = \frac{1}{2}[X_m^T, X_n^T].\begin{bmatrix} L_m & B \\ B^T & L_n \end{bmatrix}.\begin{bmatrix} X_m \\ X_n \end{bmatrix} = \frac{1}{2}(X_n^T.L_n.X_n + X_m^T.L_m.X_m + 2X_m^T.B.X_n) \quad (1)$$

Étant donné X_m, la minimisation de (1) est de résoudre le système linéaire :

$$\boxed{L_n X_n = -B^T X_m}$$

Ici, on explicite la taille de L_n, X_n : Si l'on note N le nombre des sommets sans label. M le nombre des points masqués. Donc $X_n \in \mathbb{R}^N$ et $L_n \in \mathbb{R}^{N\times N}$. Également, $B \in \mathbb{R}^{M\times N}$, $L_m \in \mathbb{R}^{M\times M}$. Avec $M + N = W \times H$

3.1.2 Résolution d'un problème $Ax = b$

Pour l'application de cette méthodes dans le domaine médicale, la taille d'une image dépasse facilement 128×128 (la taille des images dans notre expérience). En pratique, les points

non marqués sont de même ordre de grandeur de $M \times N$ $\mathcal{O}(1.6 \times 10^4)$. Pour stoker la matrice laplacienne, une occupation du mémoire sera donc à l'ordre $\mathcal{O}(3 \times 10^8)$. La décomposition LU serait pas une méthode efficace comme la solution bien que cela nous donne une solution exacte. Nous sollicitons donc à la méthode itérative. Dans notre code, nous avons implémenté la méthode gradient à pas optimal, à pas fixe et l'algorithme du gradient conjugué[3].

Ici, on présente l'algorithme du gradient conjugué, ce qu'on a utilisé dans notre expérience et qui converge plus vite parmi les trois méthodes.

Gradient Conjugué	- Choisir X^0 un vecteur initial et on pose $r^0 = -B^T X_m - L_n X^0$ et $p^0 = r^0$ - Itérer pour n$\geq$0 : $X^{n+1} = X^n + \alpha^n p^n$ avec $\alpha^n = \frac{<r^n, r^n>}{<L_n p^n, p^n>}$ $r^{n+1} = r^n - \alpha^n L_n p^n$, $p^{n+1} = r^{n+1} + \beta^n p^n$ avec $\beta^n = \frac{<r^{n+1}, r^{n+1}>}{<r^n, r^n>}$

Algorithm 1 – Gradient Conjugué

3.2 AtlasNet

Dans l'article [2] de Vakalopoulou, on a introduit une schéma pour la segmentation d'image faute de données suffisamment pour entraîner un réseaux de neurone très profond. Le modèle introduit s'appelle *AtlasNet* [2], qui applique aux images à segmenter des transformations non-linéaire mais inversible avant de les passer dans le réseaux *SegNet* [4]. Après l'entraînement, *AtlasNet* regroupe les segmentations sorties à l'issue des transformation inversées par un vote.

3.2.1 SegNet : Une segmentation par Encoder-Décoder

SegNet est un réseau convolutionnel profond, il encode et décode l'image pour obtenir une classification par pixel, dit la segmentation 'pixel-wise'. Cette architecture est proposée par V.Badrinarayanan, A.Kendall et R.Cipolla en 2016. Les codes sont publique et écrit par *Caffe*.

Cette architecture a pour le but d'entraîner un réseau très profond. Ayant la même difficulté lors la classification d'une image par *Fully Connected Layer*, il est impossible et inutile d'utiliser un réseau de neurone sans pooling. D'une part, nous pouvons retirer les caractéristiques d'une image par pooling, d'autre part, le coût d'entraînement est remarquablement réduit par pooling (sub-sampling). On peut réduire le nombre des paramètres de 134M à 14.7M par ce *sub-sampling*[4].

Très proche d'une architecture de VGG16 [5], chaque *encoder* couche effectue une convolution avec un filtre de taille 3×3, *stride* étant 1. Il conserve la forme en prenant un *padding* de longueur 1. Chaque couche est suivie par un *Batch Normalisation* qui évite la singularité du gradient, et une fonction *ReLU* qui rend la non-linéarité de ce réseau. En même temps, le nombre de *channel* devient de plus en plus grand pour tirer des caractéristique de l'image.

Max-pooling est réalisé avec une fenêtre glissante de taille 2×2 dont le *stride* est de longueur 2 qui exclut le chevauchement. Après chaque pooling, la taille de l'image se réduit par deux et devient $\frac{1}{2}H \times \frac{1}{2}W$.

Decoder couche est un processus inversé de *encoder*. Avec la même propriété pour les couche

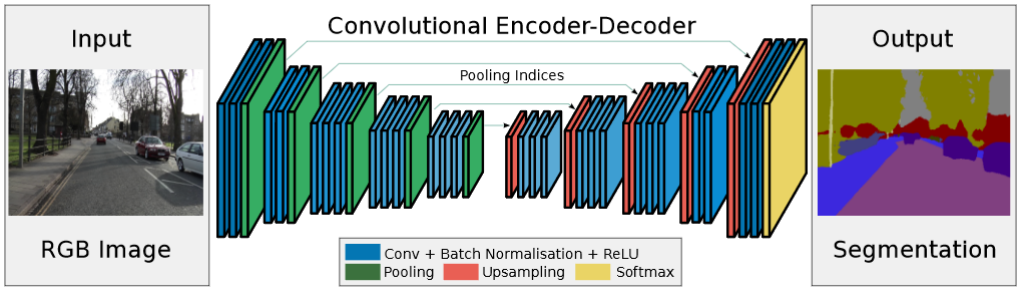


Figure 2 – Représentation de l'architecture de SegNet. Les couches cachées sont des couches convolution 2D avec Dropout. Encoder par des max-pooling et puis décoder par Upsampling. La couche sortante est un Softmax qui fournit la possibilité de classification

convolutionnelle. *Upsampling* ici est représenté par un pooling inversé, qui fait grandir la taille d'une image par 2×2. La différence entre le *upsampling* ici et celui ordinaire est que l'on remplit des pixels avec zéro sauf l'indice qu'on a utilisé pour pooling avant, à savoir, $Argmax_{2\times2}$ de l'image original.

A la fin, si l'on essaie de classifier une image en n zones différentes, on obtient une image de n *channels*. Après le traitement *softmax*, chaque *channel* représente la possibilité d'appartenir à une certaine catégorie.

3.2.2 AtlasNet : apprentissage profond et la transformation non-linéaire.

L'imagerie médicale est souvent sensible, chère et rare comme cela concerne souvent la privée des patients et que le nombre de patients sont souvent limité. Conçu en 2018 par M.Vakalopoulou *et al.*, *AtlasNet* cherche à entraîner un réseau profond sans avoir beaucoup de données.

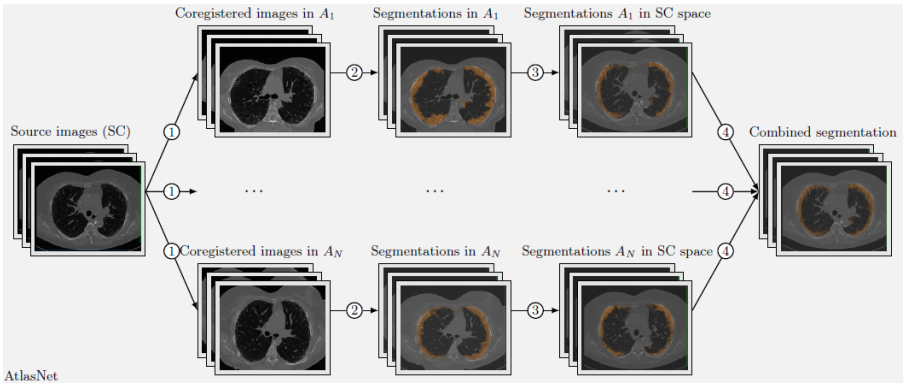


Figure 3 – Représentation de l'architecture de *AtlasNet*[2]

L'architecture de l'AtlasNet est présentée comme Fig 3.

Pour chaque piste, on entraîne un réseau *SegNet* afin d'obtenir une segmentation qui correspond à l'image transformée. De même, on applique les transformations aux images masquées. Suivant chaque direction, le but est de minimiser une fonction de perte *weighted cross entropy*

qui est définie par :

$$\sum_{i,j} -y_{i,j}\log(p_{i,j}) \times w_y - (1-y_{i,j})\log(1-p_{i,j})$$

où $y_{i,j}$ est le vrai label du pixel qui se situe à $[i,j]$ sur l'image, et $p_{i,j}$ est la probabilité prédite par le modèle. w_y est le poids pondéré sur une certaine classe y. Si $w_y > 1$, on pénalise la proportion *false négative*. Par exemple, quand on veut identifier un *tumeur*, on a une forte préférence à ne pas se tromper quand $y_{i,j}$ est un vrai tumeur plutôt que de le classifier comme un tissu normal.

La transformation inversée projette la segmentation déformée dans l'espace original.On pourra utiliser le critère de *multimétrique* [6]. Il cherche une combinaison optimale entre différentes mesures, ce qui réduit la variance de la sortie et donne une stabilité aux résultats : Prenons ρ_j, $j \in \{1,..,sk\}$, k mesures de similarité, T l'ensemble des transformations et S l'image de source et A_i un atlas (une classe de labels). Une énergie à optimiser s'écirt :

$$E(T) = \iint_\Omega \sum_{j=1}^{k} w_j \rho_j(T(S), A_i)\mathrm{d}\omega + \alpha \iint_\Omega \Psi(T)\mathrm{d}\omega$$

où, Ψ est une fonction de pénalisation pour donner une régularité sur T.

Pour le regroupement des résultats, un *Fully Connected Network* peut être utilisé comme un processus de vote pour obtenir la sortie finale.

4 Expérience

Dans le cadre de ce projet, nous avons comparé la performance de *Random Walker* et *AtlasNet*. Nous allons, dans cette partie, présenter 1.les données utilisés, 2.les détails d'implémentation qui serait util pour reproduire les résultats, 3.les résultats obtenus et une comparaison entre les deux approches.

4.1 Données

Les données sont générés dans le but de tester la performance. Les codes se trouve dans le scipts `EclipseGenerator.py` Il s'agit des ellipses déformées ou bruitées. La céation des données est basée sur la rotation et la déformation dans la coordonnée polaire.

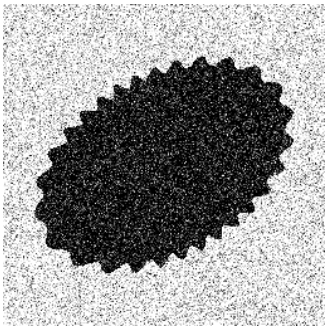


Figure 4 – Ellipse déformée - les données pour l'entraînement

Considérons une matrice carrée de taille $N \times N$, le centre de la matrix est pris comme l'origine $\mathcal{O}$. Pour le point $M_{i,j}$, posons $x = i - \lfloor \frac{N+1}{2} \rfloor, y = j - \lfloor \frac{N+1}{2} \rfloor$, alors :

$$[x', y']^T = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

représente une projection inversée avec θ l'angle avec lequel on souhaite tourner l'axe de l'ellipse. Si :

$$(\frac{x' - a \times \cos(n\phi)}{a})^2 + (\frac{y' - b \times \sin(n\phi)}{b})^2 \leq 1$$

On colore le pixel $[x, y]$, avec n le nombre de "cornes" autour de l'ellipse et $cos(\phi) = x/\sqrt{x^2 + y^2}$.

Le bruit Gaussian est ajouté par une simple sommation de la matrice originale par une autre matrice aléatoire dont les composantes sont indépendants.

$$M' = \max(1, M + \mathcal{G})$$

Par cette construction, on obtient une ellipse comme Fig 4.
Remarque : La construction par cette méthode n'a pas besoin d'interpolation, la couleur de tout pixel est bien définie.

Pour entraîner le réseau, on a créé 300 images d'ellipse de taille 128×128, avec $a, b \sim \mathcal{U}(0.1, 1)$, $\theta \sim \mathcal{U}(0, \pi)$, la variance du bruit $v \sim \mathcal{U}(0, 0.3)$ et le nombre de déformations marginales $N \sim \mathcal{U}(0, 50)$. Toutes les variables sont indépendantes.

4.2 Détails d'implémentation

Les algorithmes sont programmés ou implémentés en *Python3*. Pour l'algorithme *Random Walker*, on a utilisé le 4-voisins modèle (le 8-voisins modéle est aussi programmé dans le script `Weight.py`) et $\beta \in [50, 300]$. Pour AtlasNet, il faut *tensorflow* et *Keras* comme l'environnement requis.

L'exécution de *SegNet* est basée sur *transfer learning* : En prenant le modèle pré-entraîné (`https://github.com/alexgkendall/SegNet-Tutorial`), nous avons enlevé la couche *Softmax* qui donne la possiblité d'une segmentation de 12 classes. Nous ajoutons ensuite trois couches `Conv2D` avant de passer à *Softmax* qui finalement fournit une segmentation en 2 classes. La structure est présentée comme Figure 5.

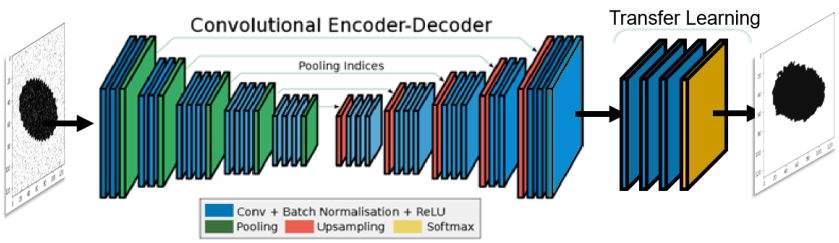


Figure 5 – Représentation de l'architecture de *tranfer learning*

4.3 Résultats

Pour entraîner *SegNet*, nous avons crée 20 images comme on n'a pas toujours beaucoup de données en imagerie médicale. En tant que *Random Walker*, il s'agit d'une méthode sans apprentissage et déterministe. Donc, le nombre de donnée n'a pas d'influence sur la performance de *Random Walker*.

On a testé les deux modèles sur 10 images générées. La précision et la valeur de fonction de perte sont présentées :

	Test accuracy	Test loss
SegNet	95.03	0.1837
Random Walker	98.70	0.1793

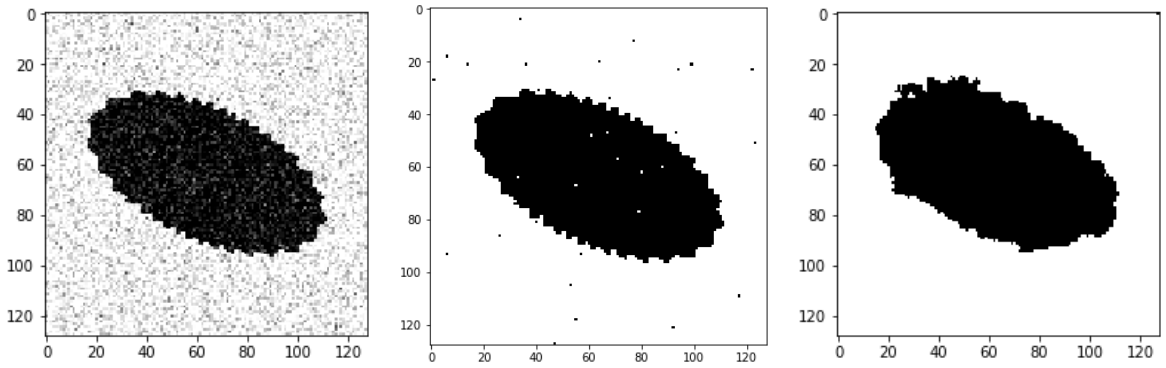
Table 1 – Précision et perte moyennes

Remarque : Effectivement, on peut améliorer la performance de *SegNet* avec plus de données. On a obtenu une précision de *97.20* avec 210 images comme données d'entrée.

Bien que la précision de *Random Walker* est plus haute dans notre exprérience, la performance dépend fortement de β (ici, nous avons pris $\beta = 150$). *SegNet*, une fois entraîné, est beaucoup plus adaptatif. Cela a répondu notre intuition car l'approche par *Machine Learning* est *data driven* et qu'il va réduire la variance des résultats.

Nous allons présenter deux cas représentatif de nos résultats :

1. *Random Walker* détecte mieux le contour :
   Dans ce cas, *Random walker* a trouvé presque toutes les subtilités du contour. Son robustesses pour détecter de différentes zones est remontré. En comparaison, *SegNet* présente une forte capacité à identifier des zones bruités sans les labels à priori.



Original image | Random walker avec $\beta = 90$ | Segnet

Figure 6 – Segmentation : Gauche : l'image originale avec bruit Gaussian et la déformation contour ; Milieu : Random walker détection avec $\beta = 90$; Droite : segmentation avec SegNet

   Pour une première comparaison, le *random walker* gagne en terme de précision et la performance pour identifier le contour. Visuellement, *Random Walker* est plus sensible au bruit : On constate que l'image détectée est encore (un peu) bruitée. Et le résultat dépend fortement à β. Une étude un peu plus profonde sera présentée dans la sous-section suivante. *SegNet*, après entraînement, présente une forte capacité à dé-bruiter l'image originale au détriment de la sensibilité à la variété du contour.

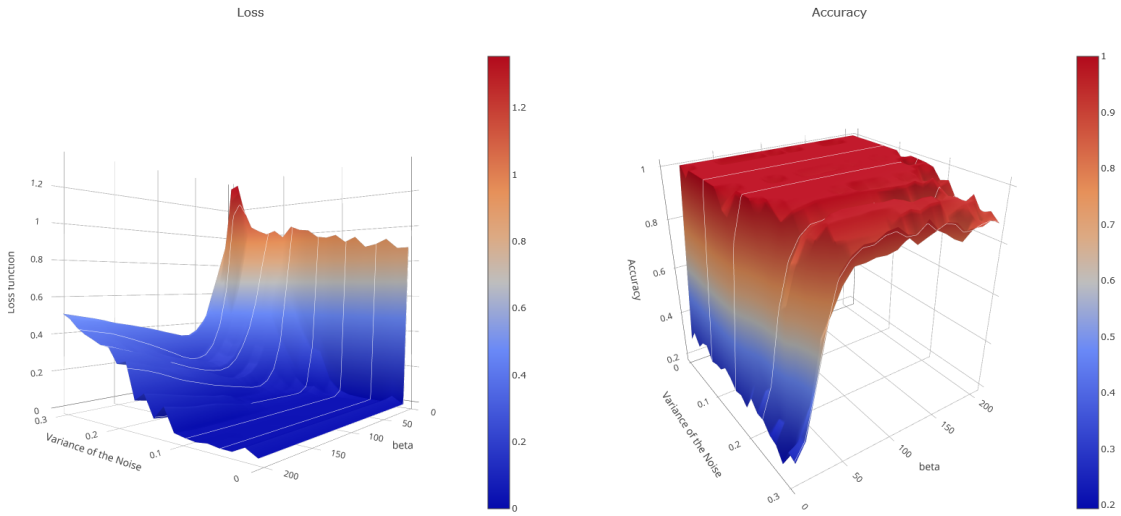
Accuracy	
SegNet	96.31
RandomWalker	99.79

Table 2 – Précision de Figure 6

Ce comportement répond à notre intuition : Bien que *Random walker* soit une méthode déterministe, le résultat représente une vecteur de probabilité de taille card(E) : à partir d'un point masqué, un *Random walker* va choisir un chemin avec une probabilité qui minimise le gradient normalisé entre le point du départ et le point d'arrivée. Donc, il s'agit d'une méthode basée sur le contraste du pixel, ce qui rendre l'approche plus sensible à l'écart du contraste. Pourtant, *SegNet*, en tant que réseau convolutionnel, dont le noyau est de taille 3×3 qui pollue automatiquement les détailles entre deux pixels, est construit pour *Pattern recognition*. Au lieu de retirer plus d'informations du contour, il cherche à maximiser le gain pour identifier un objet comme une zone particulière.

2. *Sensibilté à β - Random Walker*

   Pour *random Walker*, nous avons fait une analyse sur l'influence du β sur la précision et la fonction de perte.



Loss=$f(\beta, var)$ Accuracy=$g(\beta, var)$

Figure 7 – Loss and accuracy as a function of β and *variance*

Nous avons généré 210 images dont le bruit est de variances 0 à 0.3. Donc, chaque valeur de variance a 10 données dont la forme est indépendamment aléatoire. Pour tester β nous avons pris les valeurs de 5 à 205, linéairement distribuées. Pour chaque batch de données (10 images de même variance), on exécute *random walker* et fait évaluer la fonction de perte et la précision. Le coût de calcul est de l'ordre $\mathcal{O}(21 \times 21 \times 10) \times \mathcal{C}(Random\,Walker)$. Le temps de calcul pour cette représentation est de l'ordre 3 heures.

Avec cette présentation (Figure 7), on a deux conclusion : 1) La fonction de perte est croissante suivant la variance, ce qui est intuitif : Quand l'image est plus polluée, il est plus difficile à segmenter. 2) **La fonction de perte est expérimentalement convexe**

en β. Cela rendre possible à optimiser β même si l'on ne connaît pas la variance du bruit.

En effet, si l'on reprend la formule du gradient définie avant :

$$w_{i,j} = \exp(-\beta(g_i - g_j)^2)$$

Donc, la fonction à minimiser est **une intégrale Dirichelet** de forme ($g(x, y)$ est le contraste sur le point (x, y) :

$$\iint \exp(-\beta \operatorname{div}(g(x, y)))\mathrm{d}x\mathrm{d}y$$

Grâce à la convexité de la fonction exponentielle, on pourra minimiser cette valeur en fonction de β. Mais, la dépendance entre la fonction de perte (*Cross Entropy* qui est aussi convexe) et β est moins évidente. L'intuition est liée au fait que le minimum β^* de l'intégrale serait un candidat qui minimise la fonction *Cross Entropy*. D'où viendrait cette convexité dan le Figure 7.

5 Conclusion

L'inconvénient de *Random Walker* est immédiat : il faut manuellement spécifier les différents zones (e.x. un organe sur une imagerie de X-ray), ce qui demande souvent l'expertise en anatomie. Pourtant, son robustesse est son atout car les données des patients sont souvent sensibles et rares.

Approche par le réseau de neurone, dont le modèle ayant été entraîné, n'a plus besoin de la notation par des médecins. Pourtant, le modèle demande des données pour reprendre sa performance.

En plus, par notre expérience, *Random Walker* capture mieux les détailles alors que *SegNet* résiste mieux du bruit pixel.

Vers une amélioration :

D'un côté, vu que l'on a obtenu une relation entre β, la variance du bruit et la valeur de la fonction de perte, on pourra optimiser *Random Walker* par une approche posteriori : supposons le bruit sur chaque pixel est *i.i.d* et indépendant de du contexte de l'image. On pourra simuler la loi du bruit et prendre le meilleur *beta* qui maximise la précision.

D'un autre, inspirés par l'article de Hao-Shu Fang [7] (détection de la pose d'une personne), nous pourrons automatiser la transformation non-linéaire dans l'architecture *AtlasNet*, on pourra tester SSTN(symetrical spatial transformer network) [8] qui cherche à optimiser également la transformation de type :

$$[x^a, y^a]^T = \begin{bmatrix} \theta_{1,x} & \theta_{1,y} & \theta_{1,1} \\ \theta_{2,x} & \theta_{2,y} & \theta_{2,1} \end{bmatrix} \begin{bmatrix} x^b \\ y^b \\ 1 \end{bmatrix}$$

Où Θ est une matrice à optimiser. Il présente un fort robustesse par rapport aux transformation artificielle. À la fin, on peut se douter à la signification d'une transformation arbitraire même si cela va donner un meilleur résultat. **Doit-on faire la concession des connaissances et des expériences de l'humain devant cette boîte noire ?**

Références

[1] Leo Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(11) :1768–1783, Nov. 2006.

[2] Maria Vakalopoulou, Guillaume Chassagnon, Norbert Bus, Rafael Marini Silva, Evangelia I. Zacharaki, Marie-Pierre Revel, and Nikos Paragios. AtlasNet : Multi-atlas Non-linear Deep Networks for Medical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Granada, Spain, September 2018.

[3] J. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Available online [1]. Carnegie Mellon University, 1994.

[4] Alex Kendall, Vijay Badrinarayanan, , and Roberto Cipolla. Bayesian segnet : Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv :1511.02680*, 2015.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[6] Enzo Ferrante, Puneet K. Dokania, Rafael Marini, and Nikos Paragios. Deformable registration through learning of context-specific metric aggregation. *CoRR*, abs/1707.06263, 2017.

[7] Haoshu Fang, Shuqin Xie, and Cewu Lu. RMPE : regional multi-person pose estimation. *CoRR*, abs/1612.00137, 2016.

[8] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

♠
♢ ♡
♣

1. http ://www.cs.cmu.edu/∼./quake-papers/painless-conjugate-gradient.pdf