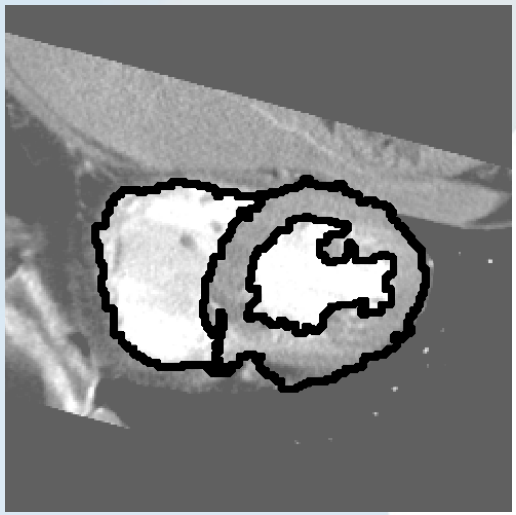
Segmentation d'images

MAP512

Te Sun Ge Jin Superviseur : Stéphanie Allassonnière

19/03/2019



Code : https://github.com/sun-te/MAP512_Segmentation/

ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

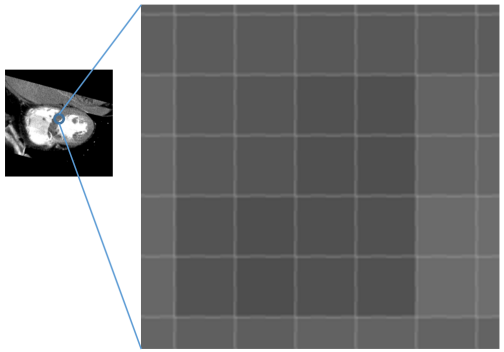
Plan de la présentation

Section 1

Méthodes de Segmentation

Random Walker : Construction du Graphe

- Une **image** peut être représentée par un **graphe** connecté



- Normaliser l'image pour que $g_i \in [0, 1]$
- g_i : **l'intensité** sur le sommet V_i
- 4-voisins modèle vs 8-voisins modèle
- pour chaque arête, donner un **poids** défini par :

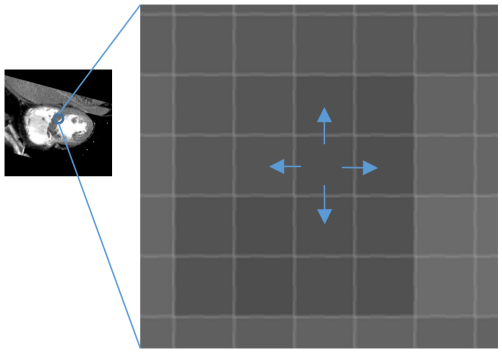
$$w_{i,j} = \exp(-\beta(g_i - g_j)^2)$$

où β est un paramètre à choisir.

- Le degré du sommet V_i est défini par : $d_i = \sum_{j \in V, j \neq i} w_{i,j}$

Random Walker : Construction du Graphe

- Une **image** peut être représentée par un **graphe** connecté



- Normaliser l'image pour que $g_i \in [0, 1]$
- g_i : **l'intensité** sur le sommet V_i
- 4-voisins modèle vs 8-voisins modèle
- pour chaque arête, donner un **poids** défini par :

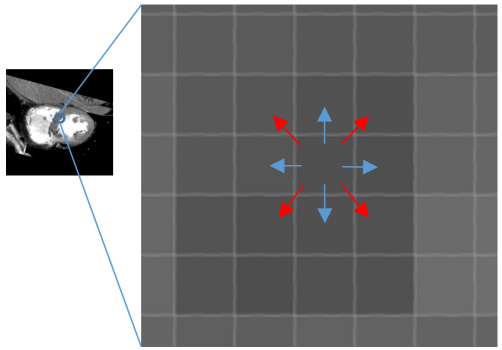
$$w_{i,j} = \exp(-\beta(g_i - g_j)^2)$$

où β est un paramètre à choisir.

- Le degré du sommet V_i est défini par : $d_i = \sum_{j \in V, j \neq i} w_{i,j}$

Random Walker : Construction du Graphe

- Une **image** peut être représentée par un **graphe** connecté



- Normaliser l'image pour que $g_i \in [0, 1]$
- g_i : **l'intensité** sur le sommet V_i
- 4-voisins modèle vs 8-voisins modèle
- pour chaque arête, donner un **poids** défini par :

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

où β est un paramètre à choisir.

- Le degré du sommet V_i est défini par : $d_i = \sum_{j \in V, j \neq i} w_{i,j}$

Random Walker : Intégrale Dirichlet

Pour un milieu continu, la segmentation est d'identifier de différentes parties qui minimisent **l'intégrale Dirichlet** :

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

Dans un milieu discret, on peut avoir une équivalence par une définition 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'intégrale Dirichelet s'exprime par :

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

Random Walker : Minimisation de $\frac{1}{2}X^T.L.X$

- $\{V_m\}_{m\in\{1,\cdots M\}}$: Les sommets avec labels ;
- $\{V_n\}_{n\in\{1,\cdots N\}}$: Les sommets non-masqués

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

La minimisation de $D[X]$ est de résoudre le système linéaire :

$$\boxed{L_nX_n = -B^TX_m}$$

Random Walker : Solution du système $Ax = b$

Dilemme de grande dimension :

- **Méthode itérative** VS. **Méthode décomposition LU**

- 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 :

$$X^{n+1} = X^n + \alpha^n p^n \text{ avec } \alpha^n = \frac{<r^n, r^n>}{<L_n p^n, p^n>}$$
$$r^{n+1} = r^n - \alpha^n L_n p^n,$$
$$p^{n+1} = r^{n+1} + \beta^n p^n \text{ avec } \beta^n = \frac{<r^{n+1}, r^{n+1}>}{<r^n, r^n>}$$

Un schéma itérative (Gradient Conjugué) pour résoudre le sytème

Remarque

Si nous avons S différents labels, nous devons résoudre S fois le système mais avec la même matrice Laplacienne

AtlasNet : architecture

AtlasNet

images ⟶ Transformations non-linéaires inversibles ⟶ **SegNet** ⟶ Transformations inversées ⟶ vote ⟶ Segmentation d'images

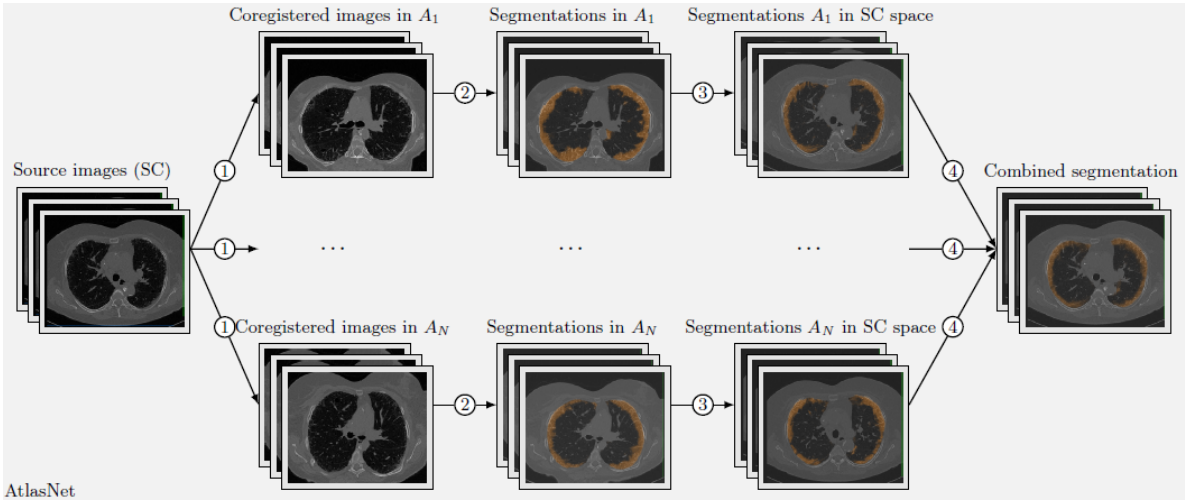


Figure 1 – Représentation de l'architecture de *AtlasNet*

AtlasNet : SegNet

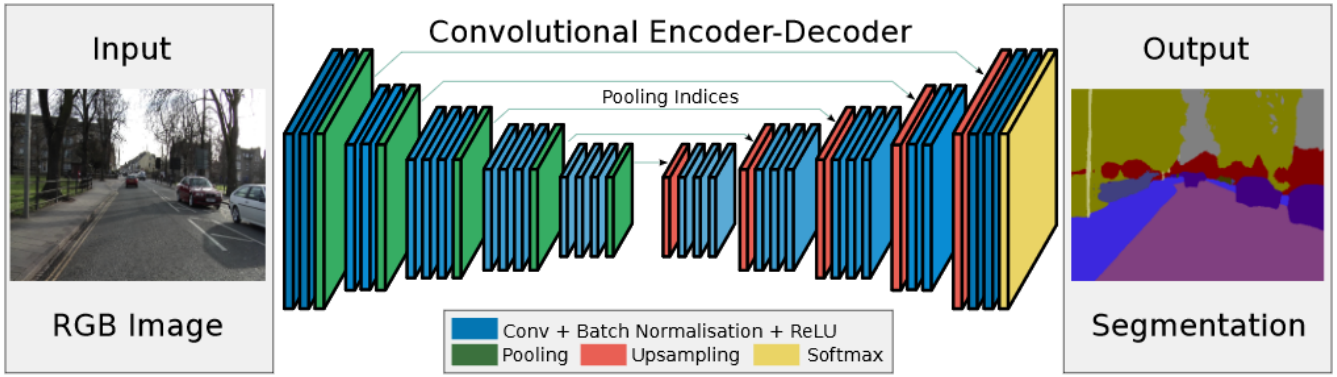
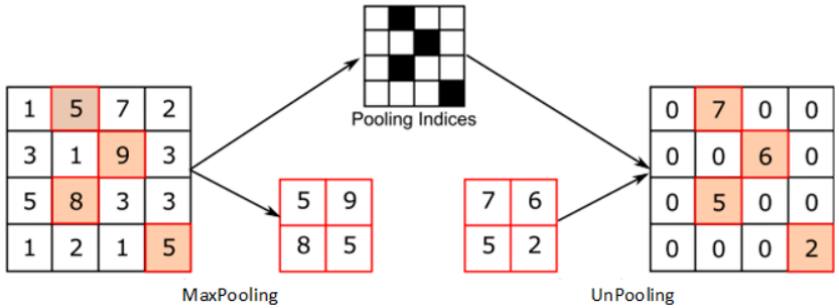


Figure 2 – Représentation de l'architecture de SegNet

Pooling & Upsampling :Concervation des pooling indices pour upsampling



AtlasNet : Synthèse

Pour chaque image tranformée, on applique *SegNet* afin d'obtenir une segmentation.

- **Loss function** :**weighted cross entropy**

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

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 prédiction par le modèle.
w_y est le poids pondéré sur une certaine classe y. Si $w_y > 1$, on pénalise la proportion ***false négative***.

Section 2

Expériences et résultats

Données : une description de dataset (1/2)

Training set : 210 images comme ci-dessous
Annotation train : 210 images image sans bruits en 2 classes (noir et blanche)
Validation & Test set : 10 images avec leurs labels.

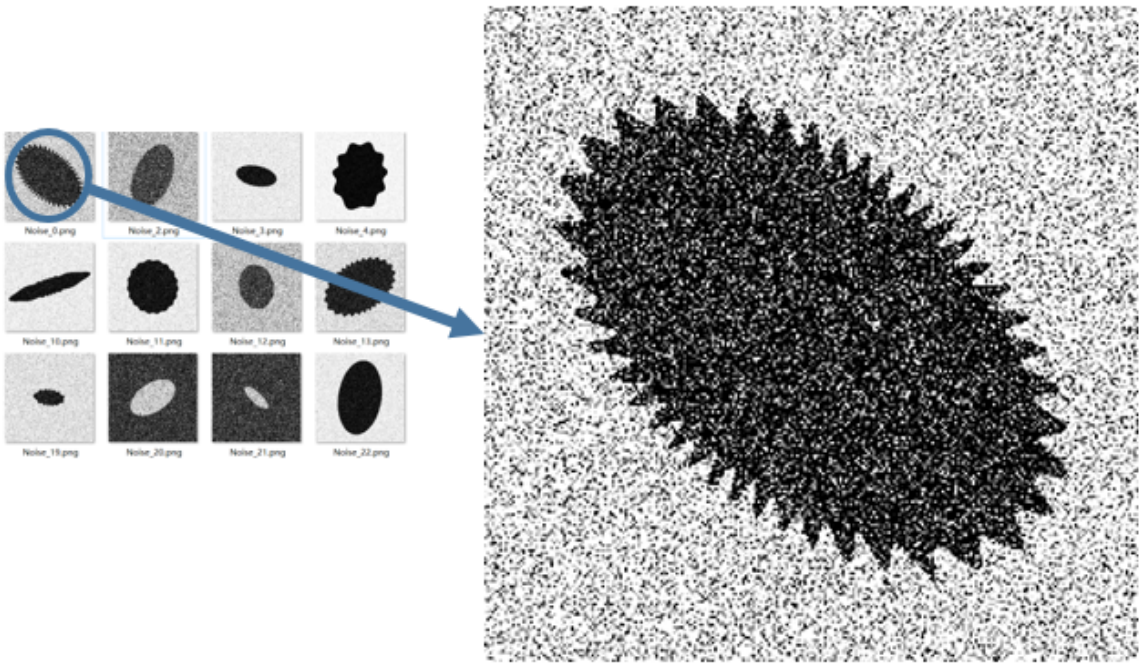


Figure 3 – Ellipse déformée - les données pour l'entraînement

Données : une description de dataset (2/2)

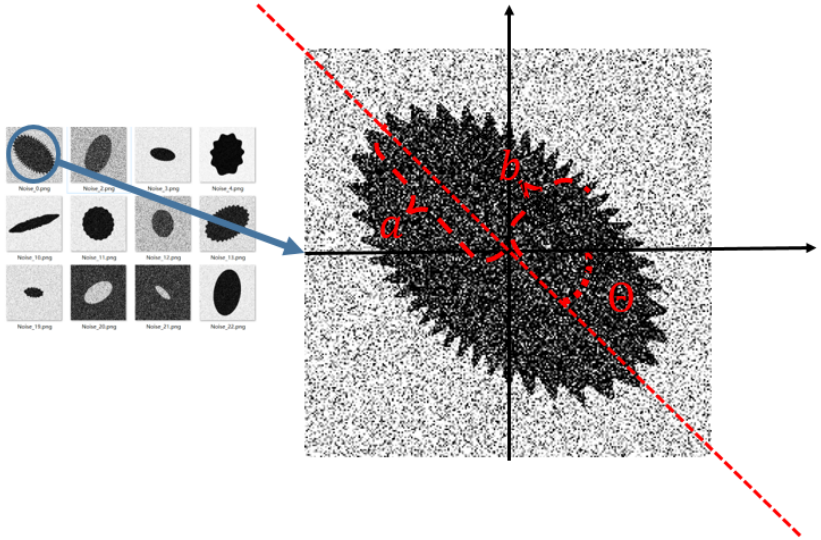


Figure 4 – Ellipse déformée - les données pour l'entraînement

- Training set (toutes les variables présentes sont indépendantes) :
  - 210 images d'ellipses déformées
  - **Images :** 128×128 pixels
  - **Ellipse :** $a, b \sim \mathcal{U}(0.1, 1) \times 64$ pixels
  - **Bruit sur pixel :** $b \sim \mathcal{N}(0, e^2)$, où $e \sim \mathcal{U}(0, 0.3)$
  - **Nombre de "Cornes" :** $N \sim \mathcal{U}(0, 50)$

Données : Comment on les a crées

1. **Centrer les points** :
   Pour le point $M_{i,j}$, posons $(x, y) = (i - \lfloor \frac{N+1}{2} \rfloor, = j - \lfloor \frac{N+1}{2} \rfloor)$
2. **Rotation** :

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

3. **Colorisation** : On colore tout pixel qui vérifie :

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

   avec avec n : le nombre de "cornes" autour de l'ellipse et $cos(\phi) = x/\sqrt{x^2 + y^2}$
4. **"Polluer" l'image :** Soit M la matrice de l'image :

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

   Où $\mathcal{G}$ est une matrice dont les composante sont *i.i.d* et gaussienne avec une variance aléatoire.

Détails d'implémentation : Random walker

- **Random walker** :
  1. Donner les "Seeds" et les "labels"
  2. Créer les sommets et les arêtes non orienté à partir de l'image
  3. Attribuer un poids pour chaque arêtes
  4. Reformuler la matrice Laplacienne ordonnée
  5. Pour chaque label, résoudre un système linéaire pour obtenir un vecteur de probabilité
     - Gradient Conjugé : Condition d'arrêt : le nombre de pas limité par 10^6 avec une tolérance : 10^{-6}. (script `Dirichelet.py`).

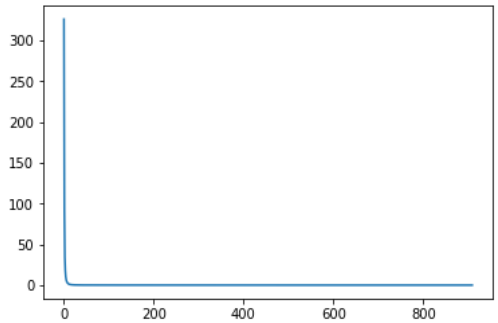


Figure 5 – L'"erreur" : $||x_{n+1} - x_n||$ - Nombre de pas

  6. Attribuer un label selon la probabilité obtenue

Détails d'implémentation : SegNet (1/2)

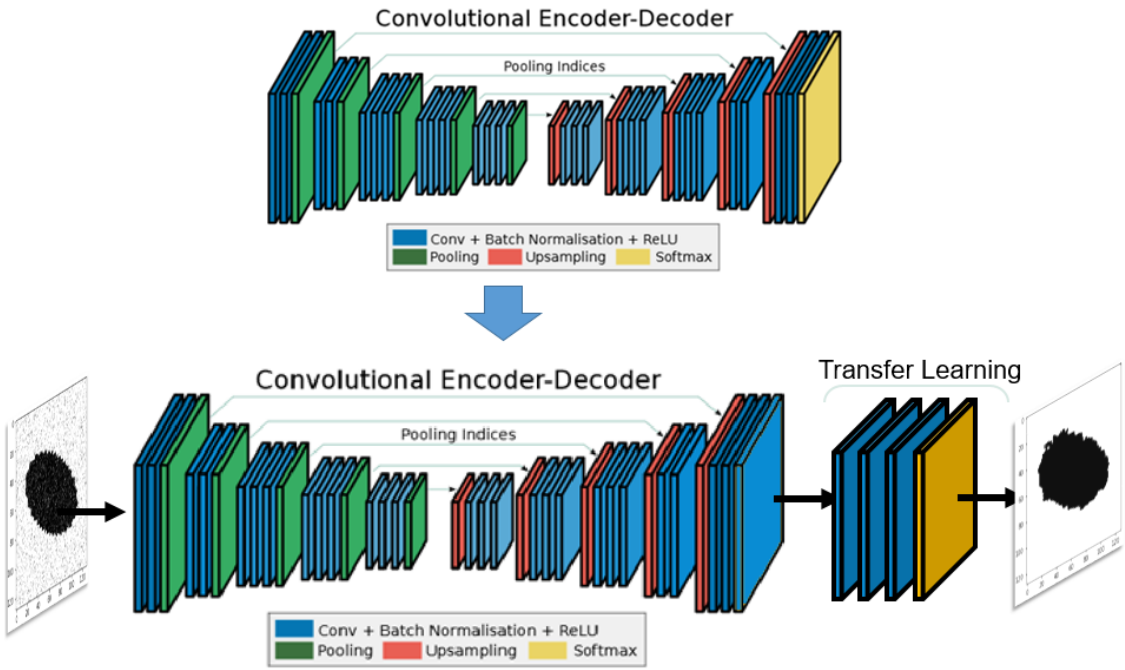
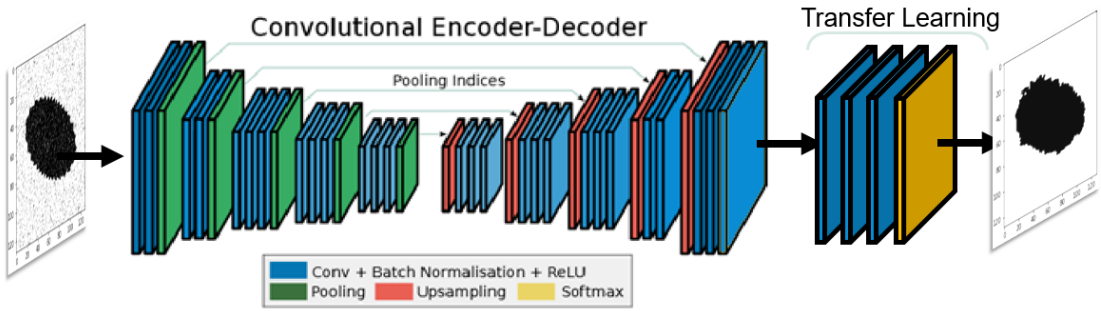


Figure 6 – Représentation de l'architecture de *tranfer learning*

Détails d'implémentation : SegNet (2/2)



- Architecture : Trois couches `Conv2D` supplémentaires avant la couche `SoftMax`
- Training :
  1. Conserver la partie *Encoder-décoder* pré-entraîner et optimiser les couches ajoutées.
  2. à partir de 200 époch, libérer la partie *décoder*.

Résultats : (description globale)

Modèles utilisés

- *SegNet* : Après entraînement sur **20** images.
- *Random walker* : Deux "seeds" : le centre et le point (127,127). Résolution avec $\beta = 150$.

Précision & Loss function

	Test accuracy	Test loss
SegNet	95.03	0.1837
Random Walker	98.70	0.1793

Table 1 – Précision et perte moyennes

Remarque :

- On peut booster 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 expérience, la performance dépend fortement de β

Résultats : (Test sur un exemple)

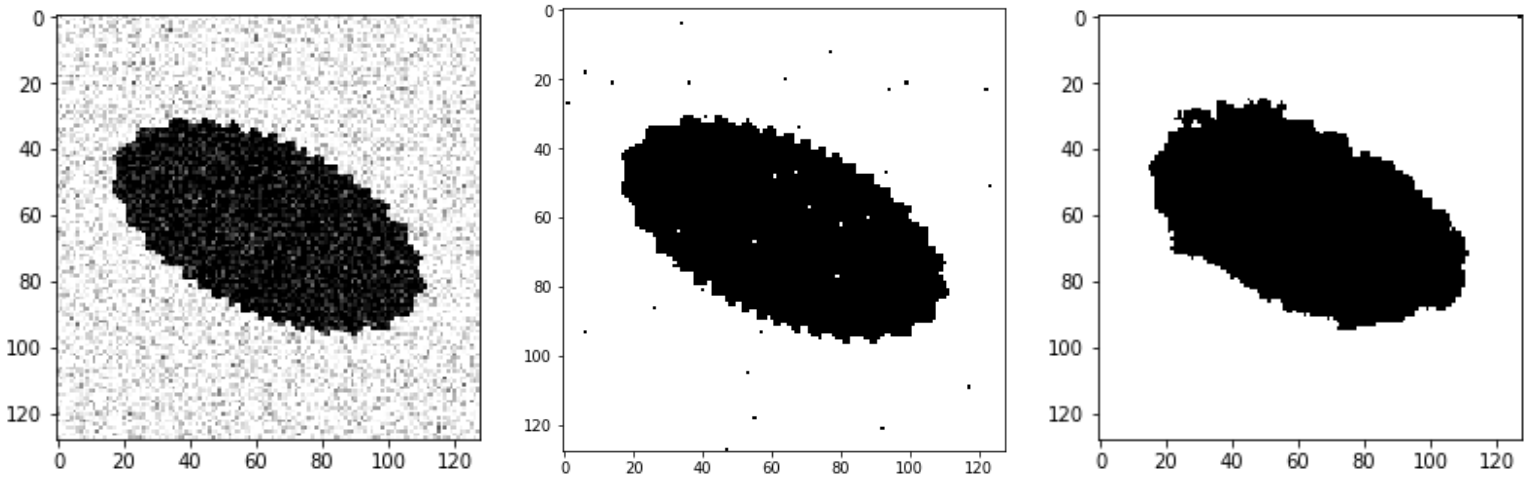


Figure 7 – Gauche : l'image originale avec bruit Gaussian et la déformation contour ; Milieu : Random walker détection avec $\beta = 90$; Droite : segmentation avec SegNet

Accuracy	
SegNet	96.31
RandomWalker	99.79

Table 2 – Précision de Figure 7

- **Random Walker** : Détection du contour
- **SegNet** : Peu influencé par le bruit sur pixel

Résultats : étude sur la sensibilité de *Random Walker* à β

x : Le valeur de l'écart type du bruit pixel, de 0 à 0.3

y : β de 5 à 205

z : Fonction perte & Précision

`[Source Image]` :Loss ; Precision

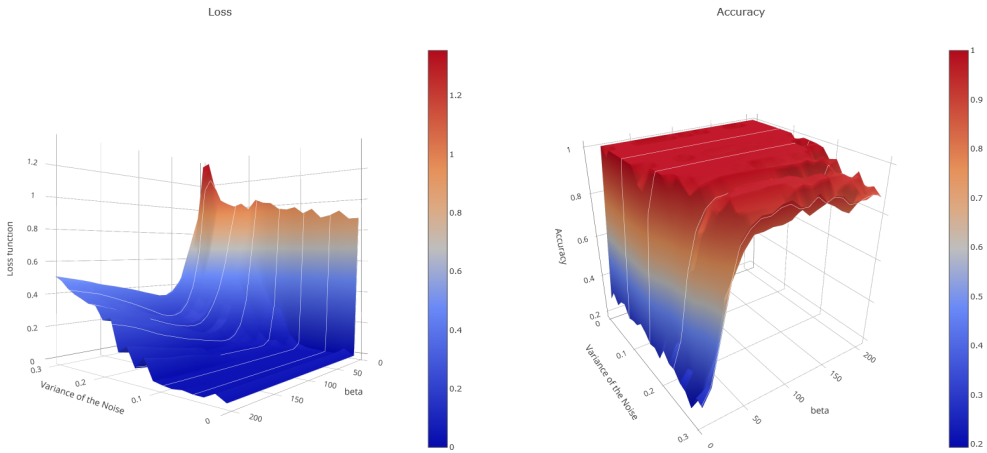


Figure 8 – Loss and accuracy as a function of β and Standard Deviation

- Une concexivité en β observée, l'optimisation possible

Section 3

Conclusion

Conclusion

Random Walker :

1. **Avantage :** Son robustesse vient de non-apprentissage. Nous pouvons l'appliquer dans plusieurs domaines.
2. **Inconvénient :** 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.

SegNet :

1. **Annotation automatique** :si le modèle a été entraîné, on n'a plus besoin de l'annotation par des médecins.
2. **Inconvénient :** le modèle demande des données pour reprendre sa performance

Perspective

- **Optimisation de *Random Walker* à posteriori :** supposons le bruit sur chaque pixel est *i.i.d* et indépendant du contexte de l'image. On pourra simuler la loi du bruit et prendre le meilleur $beta$ qui maximise la précision.
- **Automatisation de la transformation non-linéaire dans l'architecture *AtlasNet*** : on pourra la remplacer par SSTN(symetrical spatial transformer network) qui cherche à optimiser la transformation de type (par apprentissage) :

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

?

Doit-on faire la concession des connaissances et des expériences devant cette boîte noire ?

Référence

[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.

Merci pour votre attention !