Thomas Dargent,

Sous la tutelle de Caroline Petitjean

PRIOR-AWARE IMAGE SEGMENTATION

USING PRIOR KNOWLEDGE IN WEAKLY SUPERVISED LEARNING SCENARIOS

Mai 2020

UNIVERSITÉ DE ROUEN NORMANDIE

Table des matières

Merci à C. Petitjean de m'avoir tutoré sur ce projet.

Merci à Y. Zhou pour ses explications sur son article.

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

1
Introduction

La segmentation sémantique est un problème d'apprentissage qui cherche à partitionner une image en sous-zones selon leurs sens. Par exemple, localiser une personne ou un objet dans une image et délimiter la zone concernée. Ce problème n'est pas encore résolu dans beaucoup de cas, mais l'état de l'art permet d'obtenir de très bons résultats sur de multiples type d'images. On utilise en général des réseaux convolutifs profonds. La difficulté majeur de la segmentation réside dans la complexité des vérité terrain : en effet, tracer les zones correctes pour chaque image d'apprentissage est coûteux en temps. Comme on sait que les techniques d'apprentissage profonds vont généraliser mieux quand on dispose d'un grand nombre de données, cela fait également un grand nombre de vérités terrains. Beaucoup de travaux se concentrent donc sur la recherche de techniques permettant de mieux généraliser avec moins de données. Une façon de compenser ce manque est d'utiliser un savoir préalable (*Prior Knowledge*) durant l'apprentissage. Les données utilisées sont connues, et on peut potentiellement exprimer quelques règles mathématiques sur la manière dont elles sont structurées (Leurs distributions, tailles, emplacements, relations...). L'idée est que ces règles étant déterminées au préalable, elle n'auront pas à être découvertes par le modèle dans les structures de données. Ce savoir va en général accélérer l'entraînement, voir le rendre plus performant.

Dans ce rapport, on explorera divers moyens d'intégrer des connaissances préalables dans un entraînement de réseaux convolutifs profonds. Dans un premier temps on verra comment exprimer ces contraintes en utilisant des notions géométriques. On abordera ensuite différentes manière d'entraîner un réseau de neurones convolutifs en utilisant des contraintes préalablement choisies. Enfin, on présentera notre implémentation d'une méthode d'optimisation alternée. On conclura en synthétisant nos résultats, et en proposant des pistes de recherches.

2
Contraintes géométrique

2.1 Définition

Les images que l'on cherche à décrire sont sous la forme de matrice d'intensités. On aimerais pouvoir décrire la géométrie des formes avec quelques indicateurs et les donner au modèle sous forme de contrainte. Ces indicateurs peuvent s'exprimer de bien des manières mais on va en donner quelques exemples.

On définit l'espace de l'image $X \in [0,1]^N$ où N est le nombre de pixels. On cherche à segmenter les objets de l'ensemble $\mathcal{L}$. En général, un modèle prédictif de segmentation va retourner une sortie softmax $s_\theta \in [0,1]^{|\mathcal{L}| \times N}$, indiquant la probabilité pour chaque pixel d'appartenir à une classe. Chacun des pixel p dans X correspond à un ensemble de coordonnée $c_p \in \mathbb{R}^2$.

— Soft Size : Si on a une image dont tout les pixels d'un objet sont à 1 et le reste est à 0, on peut obtenir la taille en pixels de cet objet en faisant la somme des valeurs de cette image. Ici les valeurs sont des probabilités comprises entre 0 et 1, on peut donc estimer que la somme de ces probabilités nous donne une idée de la taille de l'objet. On définit donc :

$$\mathcal{S}_\theta := \sum_{p \in X} s_\theta^{l,p}, \quad \forall l \in \mathcal{L}$$

— Soft Centroid : Un centroïde peut être décrit comme la coordonnée moyenne des pixels de l'objet. Encore une fois, dans un cas binaire, il suffit de multiplier la matrice des coordonnées par la matrice des pixels, de sommer chaque coordonnée et de les diviser par la taille de la zone en question. On peut donc définir un centroïde pour nos s_θ comme étant :

$$\mathcal{G}_\theta := \frac{\sum_{p \in X} c_p s_\theta^{l,p}}{\mathcal{S}_\theta}, \quad \forall l \in \mathcal{L}$$

On peut également se servir de ces indicateurs pour en construire de plus complexes établissant des liens entre les classes :

— Ratio de tailles :

$$\mathcal{S}'_\theta := \frac{S_\theta^{l_1}}{S_\theta^{l_2}}, \quad \forall l_1, l_2 \in \mathcal{L}$$

— Relation entre centroïdes : on peut exprimer l'écart entre deux centroïdes en en faisant la différence. (On pourrait également utiliser une distance comme indicateur selon les cas)

$$\mathcal{G}'_\theta := \mathcal{G}^{l_1}_\theta - \mathcal{G}^{l_2}_\theta, \quad \forall l_1, l_2 \in \mathcal{L}$$

Comment utiliser ces indicateurs ? On va pouvoir calculer leurs valeurs sur le jeu d'entraînement et tenter de contraindre le modèle à s'en approcher. Par exemple pour la taille, on peut dire au modèle que la taille de la zone prédite doit être bornée par des limites qu'on aura déterminée [1]. On pourrait aussi lui dire que l'écart entre deux tel objet doivent garder le même signe, ou être proche d'une valeur particulière. Cela va dépendre des données que l'on cherche à prédire, c'est à l'expert de déterminer quelles contraintes vont être pertinentes. Contrôler le modèle pour qu'il respecte ces contraintes n'est pas trivial, c'est le domaine de l'optimisation sous contrainte. Il existe de nombreuse méthode (Lagrangiens, Gradient projeté...). On s'attardera ici surtout sur l'usage de pénalités. Lors de l'apprentissage les méthode d'optimisation cherche a minimiser un coût. En général, il s'agit d'une mesure de la divergence entre nos données d'entrée et la vérité terrain :

1. Kervadec, H., Dolz, J., Tang, M., Granger, E., Boykov, Y., and Ayed], I. B. (2019). Constrained-cnn losses for weakly supervised segmentation. *Medical Image Analysis*, 54:88 – 99

$$L(\Theta, X, Y) = C(\Theta(X), Y)$$

Avec Θ notre modèle, L notre fonction de coût, X et Y une image d'entrée et sa vérité terrain respectivement, et C une mesure de divergence (Entropie croisée, Indice de Jaccard, Coefficient de Dice...). Toutefois on pourrait chercher à minimiser une autre métrique qui est exprimée à l'aide des formules vues plus haut. Reprenons l'idée d'une mesure de taille : On a mesuré une taille moyenne $\bar{\mathcal{S}}_\theta$ pour chaque objets des vérités terrain d'entraînements. on peut imposer imposer au modèle à travers la fonction de coût de pas trop s'en éloigner :

$$L(\Theta, X, Y) = C(\Theta(X), Y) + \lambda \| \bar{\mathcal{S}}_\theta - \mathcal{S}_\theta(\Theta(X)) \|$$

Si la taille est égale à la moyenne, alors la pénalité est nulle, la condition est remplie. On vois bien l'intérêt pour des annotations peu fiables (car non présentes ou partielles par exemple), la pénalité ne tiens pas compte de la vérité terrain. Cette condition reste cependant très basique. Le problème de l'ajout de contraintes par pénalité est qu'il n'y a aucune garantie d'arriver à l'optimalité de la solution. De plus l'ajout de multiples pénalités peut créer des conflits entre celles à résoudres et rendre l'entraînement instable. La méthode reste toutefois très pratique pour de l'apprentissage profond grâce à sa simplicité.

3
Réalisation

3.1 Référence

On a cherché a implémenté l'article "Prior-Aware Neural Network for Partially Supervised Multi Organ segmentation" [1]. Cet article cherche à répondre au problème du manque de données dans les dataset médicaux d'une manière très intuitive. On dispose d'un grand nombre de jeux de données médicaux annotées. Il s'agit parfois de quelques organes seulement, parfois de tous. Si l'on souhaite produire une prédiction d'un ensemble d'organe, il semble qu'utiliser des jeux de données annotant même qu'une partie de ces organes permet d'avoir accès à beaucoup de données aisément. Cette idée est d'autant plus plaisante que les données médicales sont en générale très standardisées, et similaires (on a toujours les mêmes organes à peu près aux mêmes endroits, à la même taille). Pour guider l'entraînement, une pénalité est ajoutée à la fonction objective. Il s'agit de minimiser la divergence entre la distribution des organes dans notre dataset entièrement supervisées, et celle prédite pour les dataset partiels.

1. Zhou, Y., Li, Z., Bai, S., Wang, C., Chen, X., Han, M., Fishman, E., and Yuille, A. (2019). Prior-aware neural network for partially-supervised multi-organ segmentation

Afin de se familiariser avec le type de problème proposé ici, on commencera par présenter un jeu de donnée artificiel relativement simple. On cherchera ensuite a comprendre comment se déroule l'entraînement dans cet article, et on terminera en présentant le travail réalisé et nos résultats.

3.2 Jeux de données

L'article original utilise le jeu de données MICCAI 2015 Abdomen. Il contient 30 scans (tomodensitométrie) avec 3779 images au total. Les organes annotés sont les 13 suivants : La rate, les reins (gauche et droit), la vésicule biliaire, l'œsophage, le foie, l'estomac, l'aorte, la veine cave inférieure, la veine splénique, le pancreas, et les glandes surrénales (gauche et droite). Les images ont une résolution de 512×512 pixels. Pour compléter ces données entièrement supervisées, les auteurs ont utilisés trois autres sources de données :

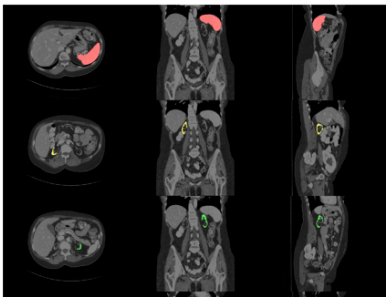

Figure 3.1: Exemples de données issues du jeu de donnée MICCAI 2015

— Un jeu de donnée annotant la rate de MedicalDecathlon
— Un jeu de donnée annotant le foie de MedicalDecathlon

— Un jeu de donnée annotant le pancreas : Pancreas-CT par TCIA

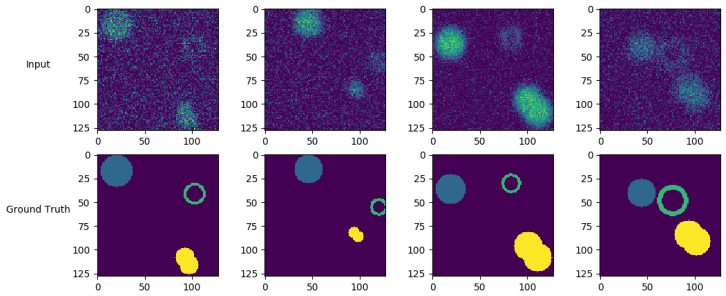

Figure 3.2: Quelques exemples d'images de notre jeu de donnée artificiel. En haut les entrée, et en bas les annotations associées.

Afin d'accélérer le cycle de développement, on a choisi d'utiliser un jeu de donnée jouet. On a généré un jeu de donnée artificiel comprenant trois objets annotés. Ils sont placés dans des zones définies mais leurs centres varient :

— En haut à gauche, un disque de diamètre variable

— En haut à droite, un cercle de diamètre et d'épaisseur variable

— En bas à droite, deux disques superposés en léger décalage. Le décalage, et le diamètre des cercles varie.

Cela nous permet de simuler des images annotées d'organes : les objets sont positionnés dans des zones déterminées, avec une taille variant entre deux bornes. Pour chacun de ces objets on a également généré un jeu de donnée dont les images sont similaires au premier, mais dont seul l'objet en question est annoté. On a donc, comme dans l'article, un total de 4 jeux de données : Un jeu entièrement supervisé, et trois partiellement supervisés. Bien entendu ces données sont simples. On a donc ajouté un bruit Gaussien additif (d'écart type variable) et un flou Gaussien (variable pour chaque axe).

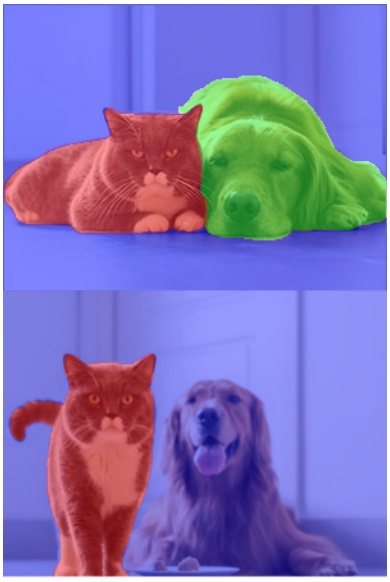

Figure 3.3: L'image du haut montre un jeu de données entièrement supervisé pour un problème à trois classes (Fond, chat et chien). Celle du bas n'est annoté que pour deux classes (chat et indéterminé). Pour avoir un score parfait sur cette image, le modèle ne doit pas prédire de chien. Cela risque d'handicaper son apprentissage général.

3.3 Entraînement

3.3.1 Principe

On se propose ici d'expliquer comment PaNN fonctionne. On a vu précédemment que leur but est d'exploiter des jeux de données partiellement annotés en parallèle d'un jeu entièrement supervisé. Une manière de le faire est d'entraîner un modèle sur tout les types d'annotation de manière indifférenciée. Le problème de cette méthode est que pour un jeu de donnée partiellement supervisé (Dans ce cadre cela signifie que tout les objets ne sont pas annotés) les objets non annotés sont traités comme un fond vide, et cela induit en erreur le modèle 3.3.

Pour palier à cela, on peut décomposer l'entraînement en deux phases : premièrement, on entraîne un modèle initial sur l'ensemble des données, puis on prédit

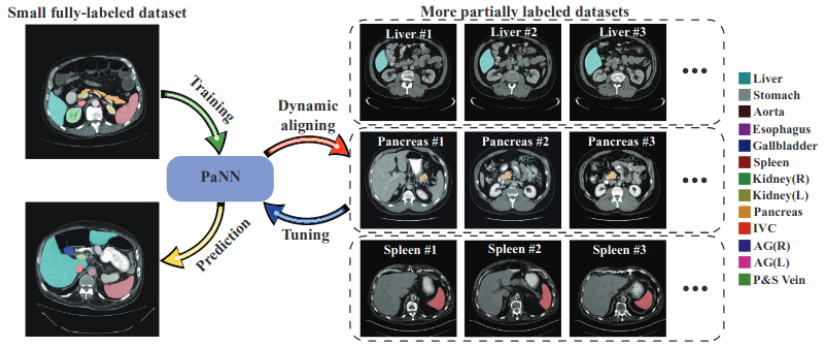

Figure 3.4: Schéma expliquant le déroulement de l'entrainement de PaNN. Source : (Zhou et al., 2019)

les annotations pour les objets non annotés dans un jeu de données, et on réentraine le modèle en utilisant ces annotations. Pour peu que les prédictions initiale soit assez bonnes, le modèle devrait s'améliorer. Toutefois, il risque également d'apprendre les défauts de ces "pseudo-labels" et de les renforcer au fur et a mesure de l'entraînement. Cela a été étudié dans certains articles. [2]

La proposition de PaNN réside en l'ajout d'un savoir préalable pour stabiliser cet entraînement. On sais que les images médicales sont très contraintes, et cela est un avantage. Les auteurs supposent que la distribution des organes au sein des datasets faiblement supervisés est similaire à celle au sein du dataset entièrement supervisé. On va donc mesurer cette distribution q dans les labels du dataset avant l'entraînement, et tendant de minimiser la divergence entre celle ci et les distribution prédite pour les datasets faiblement supervisés.

On souhaite segmenter des objets de l'ensemble $\mathcal{L}$. Un jeu de donnée S contient un ensemble d'images et un ensemble d'annotations correspondantes : $S = \{X, Y\}$. On note notre jeu de données entièrement supervisé S_L, qui annote les objets dans $\mathcal{L}_L = \mathcal{L}$. Les T jeux partiellement supervisés S_{Pt}, où $0 \leq t \leq T$, annote les objets dans $\mathcal{L}_{Pt} \subseteq \mathcal{L}$. Un ensemble d'image X est composé d'images x_i, pour chaque image x_i on note les pixels qui la composent $x_{ij} \in \mathbb{R}$. On note N le nombre de pixels dans une image, soit $N = |x_i|$. Un ensemble d'annotation Y est composé de vérité terrain y_i, pour chacune d'entre elles on note sa valeur par un vecteur $y_{ij} \in \{0,1\}^{|\mathcal{L}|+1}$.

On définit $q \in \mathrm{R}^{|\mathcal{L}|+1}$, où les q^l sont les proportions de pixels annotant les objets $\mathcal{L}_l$ pour chaque Y_{Li} dans S_L. q^0 est la proportion de fond en moyenne.

$$q = \frac{1}{|I_L| \times N} \sum_{i=0}^{|I_L|} \sum_{j=0}^{N} y_{L,ij}$$

On cherche à entraîner un modèle Θ prenant en entrée une image x_i, et retournant une prédiction p_i. Pour chaque x_{ij}, $p_{ij} \in \mathbb{R}^{|\mathcal{L}|+1}$. On note P_L et P_{Pt} les ensembles de prédictions pour l'ensemble S_L entièrement annoté et les ensembles S_{Pt} partiellement annotés respectivement.

On définit $\bar{p} \in \mathrm{R}^{|\mathcal{L}|+1}$, où les $\bar{p}^l$ sont les moyennes des probabilités d'appar-

2. Kervadec, H., Dolz, J., Wang, S., Granger, E., and Ayed, I. B. (2020). Bounding boxes for weakly supervised segmentation: Global constraints get close to full supervision

tenance a une classe $\mathcal{L}_l$ pour chaque pixels pour chaque Y_{Pt} dans un S_{Pt}.

$$\bar{p} = \frac{1}{T \times |I_L| \times N} \sum_{t=1}^{T} \sum_{i=0}^{|I_P t|} \sum_{j=0}^{N} p_{L,ij}$$

L'objectif de l'entraînement de notre modèle Θ peut être écris ainsi :

$$\min_{\Theta, P_p} \mathcal{J}_L(\Theta) + \lambda_1 \mathcal{J}_P(\Theta, P_P) + \lambda_2 \mathcal{J}_C(\Theta, q)$$

où P_P est l'ensemble des prédictions du modèle pour les datasets S_{Pt}. $\lambda 1$ et $\lambda 2$ sont des hyper-paramètres. Tentons de décrire chacun des membres de cette formule :

— $\mathcal{J}_L(\Theta)$ est l'entropie croisée calculée sur S_L. Soit :

$$\mathcal{J}_L(\Theta) = -\frac{1}{|I_L| \times N} \sum_{i=0}^{|I_L|} \sum_{j=0}^{N} \sum_{l=0}^{|\mathcal{L}|} \mathbb{1}(y_{L,ij} = l) \log(p_{ij}^l)$$

— $\mathcal{J}_P(\Theta, Y_P)$ est l'entropie croisée calculée sur les S_{Pt}. On définit un ensemble de prédictions enrichies $\mathbb{Y}$ tel que pour tout les objets l définis dans Y_{Pt}, $\mathbb{Y} = Y_{Pt}^l$, et pour tout le reste $\mathbb{Y}^{\neg l} = P_{Pt}^l$.
Soit :

$$\mathcal{J}_P(\Theta, \mathbb{Y}_P) = -\frac{1}{T \times |I_{Pt}| \times N} \sum_{t=1}^{T} \sum_{i=0}^{|I_{Pt}|} \sum_{j=0}^{N} \sum_{l=0}^{|\mathcal{L}|} \mathbb{1}(\mathbb{Y}_{Pt,ij} = l) \log(p_{ij}^l)$$

— $\mathcal{J}_C(\Theta, Y_P)$ est une pénalité exprimant le savoir préalable sur la distribution des labels. Elle est égale à la divergence de Kullback-Leibler entre q et $\bar{p}$. On a :

$$KL(q|\bar{p}) = -\sum_l (q_l \log(\bar{p}^l) + (1 - q^l) \log(1 - \bar{p}^l))$$

Seulement cette fonction est très difficile à optimiser [3] :, PaNN passe donc par une estimation des termes : $\log(\bar{p}^l)) = \nu^l$ et $\log(1 - \bar{p}^l)) = \mu^l$. Ces estimations sont apprises par le modèle en optimisant le problème min-max suivant :

$$\min_{\Theta} \max_{\nu,\mu} \mathcal{J}_C = \min_{\Theta} \max_{\nu,\mu} \sum_l [(q^l \nu^l - (1 - q^l)\mu^l)\bar{p}^l + q^l \log(-\nu^l)] + \sum_l [(1 - q^l)(\mu^l + \log(-\mu^l))]$$

On peut observer que cette fonction objective pose deux problèmes majeurs. Premièrement, elle est calculée sur l'ensemble des images de l'ensemble des datasets. Cela est évidemment impossible en termes de charge mémoire, on estimera donc qu'un batch est équivalent à l'ensemble [4]. Ainsi $\bar{p}$ est l'estimation de la distribution pour ce batch seulement. Cela implique d'avoir une taille de batch suffisamment conséquente. Elle est également calculée sur les batch de tout les dataset en même temps, ce qui implique d'effectuer plusieurs prédiction avant son calcul. Deuxièmement, $\mathcal{J}_C$ introduit la nécessite d'une optimisation min-max, ce qui

3. Cela est détaillé dans
Zhou, Y., Li, Z., Bai, S., Wang, C., Chen, X., Han, M., Fishman, E., and Yuille, A. (2019). Prior-aware neural network for partially-supervised multi-organ segmentation

4. Si cela n'est pas dit explicitement dans l'article, ça a été confirmé par l'autrice

va compliquer l'algorithme d'apprentissage. Au final les auteurs proposent l'algorithme 1.

Entrées :
Un jeu de données entièrement supervisé S_L;
T datasets partiellement supervisés S_{Pt};
Les hyper-paramètres λ_1, λ_2;
Sorties :
Un modèle de segmentation sémantique Θ;
début
  Entraîner le modèle Θ sur S_L pour M1 itérations;
  répéter
    Mettre à jour les pseudo-labels P_p;
    Mettre à jour ν et μ par montée de gradient stochastique;
    Mettre à jour Θ par descente de gradient stochastique;
  jusqu'à *ce que M2 itérations soit faites*;
fin
retourner Θ

Algorithme 1 : Procédure d'entraînement

3.3.2 Expériences

Le code n'était pas fournis par les auteurs. On a donc commencé une ré-implementation sous PyTorch [5]. De par la nature complexe et inhabituel de l'entraînement, il nous a semblés plus aisé de partir de zéro que de tenter d'adapter des codes déja existant a cette méthode. On a donc codé de bout en bout un système fonctionnel, qui est entièrement disponible sur Github https ://github.com/Kalwing/PaNN. Devant l'ampleur de la tache, on a attaqué le problème en morceaux. Premièrement on s'est assuré d'avoir un modèle fonctionnel de segmentation entièrement supervisée. Pour pouvoir tester le code directement dans l'environnement de développement on a du faire quelques choix. On a utilisé les données artificielle décrite en 3.2, en les limitant à 128×128 pixels. Le réseau classique en segmentation sémantique est le U-Net [6], mais il est relativement profond et donc coûteux à entraîner. On a proposé d'utiliser une version plus légère que l'on nommera U-Net-light. Il s'agit du même réseau mais avec moins de couches **??**.

5.

6.

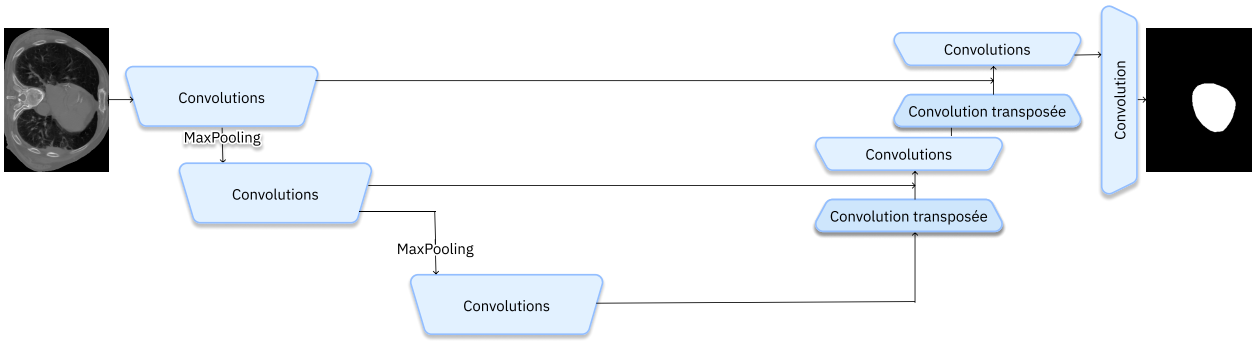

Figure 3.5: UNet-Light, le réseau utilisé pour nos expérimentations. Les blocs "convolutions" signales la présence de deux convolutions enchaînées.

Après quelques résultats satisfaisant (Fig. 3.6), on a pu essayer d'ajouter des paramètres au réseau en dehors de l'API classique de PyTorch. En effet, μ et ν doivent pouvoir être initialisé manuellement, et être mis à jour uniquement dans les phases de montée de gradient stochastique. On a intégré directement au U-Net-light ces paramètres, et les fonctions permettant de les contrôler. Ces fonctions permettent donc notamment de passer d'un apprentissage de Θ à un apprentissage de ν, μ. Ces pré-requis de principes remplis, on a pu construire PaNN.

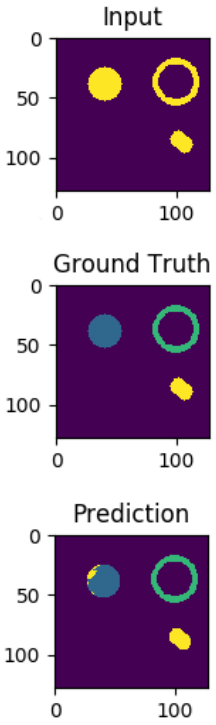

Figure 3.6: Résultats préliminaire avec une loss DICE classique, sur 20 Epochs

Le plus gros noeud de complexité est la manière de charger les données en mémoires. PyTorch fonctionnent avec des *Dataloader*. Ces objets retournent à chaque itérations de la boucle d'entraînement un batch d'un dataset et, quand ils n'y a plus de batch à retourner, on recommence pour une autre epoch. Ici on veux charger des batchs pour des datasets differents à la même itérations. Ces datasets n'ont pas la même tailles (dans l'article de référence $|S_L| = 30$ et $|S_{P_t}| = 40$), et le même nombres d'image de chaque ne sont par chargées en mémoire au même moment (Ils utilisent un ratio de 3 images de S_L pour 1 de chaque S_{P_t}). Ainsi on voit que chaque Dataloader aura terminé de parcourir ses données après un nombre diffèrent d'itération. Les auteurs semblent avoir choisi de ne pas marquer de fin d'epoch puisqu'ils parlent dans l'article d'itérations (Et cela est confirmé par l'auteur). On suppose que quand un Dataloader à terminé, il recommence, peu importe si les autres n'ont pas fini. L'entraînement se termine après un nombre fixe d'itération. Le système d'epoch est très pratique puisqu'il permet de bien marquer la fin d'une séquence d'entraînement, de regarder les scores en validation, et de sauvegarder les résultats. De plus on a compris l'intérêt du passage en itération qu'après avoir codé une base valide utilisant les epochs. On a donc fait le choix d'utiliser les epochs en chargeant les données de manières différentes : A chaque itération on charge un batch pour chaque dataset, quand un dataset n'en a plus en réserve, il recommence. Ainsi le modèle voit chaque données au moins une fois.

On a ensuite construit la fonction de coûts progressivement. Le premier membre, $\mathcal{J}_L$, est un calcul d'entropie croisée classique, facilement implémenté. Pour le deuxième, $\mathcal{J}_P$, il a fallu utiliser des prédictions antérieure dans le calcul. Ces prédictions sont

générés à l'avance et sauvegardées comme de nouveaux jeux de données. On part sur le principe que l'ordre de chargement des images ne change pas au cours de l'entraînement, ce qui évite d'avoir a transformer en profondeur les Dataloader. En effet ainsi on peut sauvegarder les données dans l'ordre dans lesquelles elles sont données, et les recharger de la même manière. Cela est également compatible avec les augmentations de données en lignes utilisées par les auteurs. Lors du calcul de la loss, pour un dataset supervisant l'objet l, on utilise les prédictions pour la même image comme pseudo label. On garde les pseudo-labels pour tout les objets sauf l, pour lequel on garde le label annoté original. Enfin reste a faire la partie la plus cruciale, le troisième terme $\mathcal{J}_C$. Il est équivalent à la divergence de Kullback Leibler entre q, la distribution des objets sur le dataset entièrement supervisés, et p la distribution prédite. Son calcul demande d'accéder directement a des paramètres appris par le modèle, ν et μ. On a donc permis a la loss d'accéder, en plus de la sortie des sorties du réseau et leurs labels associés, aux paramètres du modèles. Cette fonction de coût demande donc à chaque itération :

— Pour chaque dataset, la prédiction pour chaque batch
— Pour chaque dataset S_{P_t}, les pseudo labels associés a ce batch
— Les paramètres ν et μ du modèle
— La distribution des objets dans S_L : q

Enfin on a pu implémenter la procédure d'entraînement (Algo. 1). On alterne entre les phases de minimisation et de maximisation en inversant la fonction de coût. PyTorch avec son calcul automatique de gradient rends le choix des paramètres à optimiser plutôt simple : il suffit de bloquer ceux que l'on souhaitent fixer.

3.4 Résultats

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 :

$$\mathrm{DSC} = \frac{2|X \cap Y|}{|X| + |Y|} \tag{3.1}$$

Les expériences sur les datasets utilisés par l'article n'ont pas été réalisées. On ne dispose donc que de résultats préliminaires sur notre jeu de données jouet. Il permet toutefois de distinguer des tendances intéressantes. On a un dataset entièrement supervisé S_L de 100 images, dont 10 en test, 40 en validation et le reste en test. Les trois autres dataset S_P ont l'annotation d'une seule classe chacun, et disposent de 40 images d'entraînement. On a aussi fait quelques expérimentation avec plus d'images dans les datasets S_P (400) ou dans le dataset S_L, sans grandes différences. Le problèmes étant trés simple cela ne semble pas affecter plus que ça les tendances d'entraînement.

La tendance générale observée est que les performances se maintiennent légèrement en dessous des performances de la phase 1 pendant quelques itérations

puis baissent en géneral. Dans aucune tentative le score initiale n'a été amelioré par l'entraînement avec les datasets partiels. Cela peut être expliqué par la tendance a propager les erreurs dans les pseudo labels a chaque nouvelle prédiction. On observe en effet une tendance dans certaines erreurs à s'accentuer au fur et a mesure des itérations.

Beaucoup de causes peuvent expliquer cela :

1. La simplicité des données : Peut être que les données sont déja prédites au mieux par le modèle initial.
2. La question du nombre d'epoch/itérations d'entraînement pour chaque étape n'est pas évidente : a quel moment interrompre la première phase ?
3. La pondération de la pénalité $\mathcal{J}_C$ est peut être trop faible pour garantir son application
4. Une erreur d'implémentation n'est pas à écarter.

4
Conclusion

Au final, on a vu un cas particulier d'application de savoir préalable pour compenser le manque de données annotées. L'article original parle de résultats de bonne qualité, que l'on a pas encore réussi a reproduire. Il nous reste toutefois énormément de piste d'exploration. On pourra d'abord tenter d'utiliser le modèle avec des données réelles, pour pouvoir vraiment jauger sa qualité. Il faudra également établir un protocole pour trouver les hyper-parametres correspondant au cas traité.

Cependant les erreurs que l'on semble percevoir sont similaire à ce qui a été étudié dans la littérature sur le sujet. On peut donc être satisfait de l'ébauche de résultat que l'on a actuellement.

Bibliographie

Kervadec, H., Dolz, J., Tang, M., Granger, E., Boykov, Y., and Ayed], I. B. (2019). Constrained-cnn losses for weakly supervised segmentation. *Medical Image Analysis*, 54 :88 – 99.

Kervadec, H., Dolz, J., Wang, S., Granger, E., and Ayed, I. B. (2020). Bounding boxes for weakly supervised segmentation : Global constraints get close to full supervision.

Zhou, Y., Li, Z., Bai, S., Wang, C., Chen, X., Han, M., Fishman, E., and Yuille, A. (2019). Prior-aware neural network for partially-supervised multi-organ segmentation.