UNIVERSITÉ DU QUÉBEC À MONTRÉAL

SEGMENTATION ET ANALYSE DE DONNÉES MICROSCOPIQUES 3D POUR LA MORPHOLOGIE CELLULAIRE DES CYTONÈMES

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

PHILIPPE LEMIEUX

JUIN 2024

REMERCIEMENTS

J’aimerais remercier d’abord Joël Lefebvre, mon directeur de recherche, de m’avoir offert l’opportunité de faire une maîtrise et pour m’avoir accompagné tout au long de celle-ci. Au-delà d’un superviseur, je tiens à le remercier de m’avoir introduit au domaine de traitement d’image et m’avoir poussé à continuer mes études dans un domaine si intéressant. Au cours de ma rédaction, c’est aussi lui qui m’a encouragé à devenir un chargé de cours. Je tiens aussi à remercier nos collaborateurs à Université de Montréal (UdeM) et mes collègues au Laboratoire d’Imagerie Numérique, Neurophotonique et Microscopie (LINUM), particulièrement Basile Rambaud et Mohamad Hawchar pour leurs aides et conseils.

J’aimerais aussi remercier mes parents et amis pour leurs supports et encouragements à travers ma rédaction. Je n’aurais jamais imaginé commencer et encore moins finir un tel projet sans eux. À travers bon temps et mauvais temps, je vous en dois tellement…

TABLE DES MATIÈRES

TABLE DES FIGURES vi

LISTE DES TABLEAUX ix

ACRONYMES x

NOTATION xii

RÉSUMÉ xiii

CHAPITRE 1 INTRODUCTION 1

1.1 Contexte 1

1.2 Motivation 2

1.3 Problématique 2

1.4 Contributions 4

1.5 Structure du mémoire 5

CHAPITRE 2 MÉTHODES ET CONCEPTS 6

2.1 Bases du traitement d'images 6

2.1.1 Convolutions 6

2.1.2 Morphologie de régions 7

2.1.3 Segmentation de régions basée sur l'intensité 10

2.2 Modèle de segmentation 11

2.2.1 Architecture d'un U-Net 12

2.3 Concepts d'optimisation de modèles 15

2.3.1 Apprentissage autosupervisé 16

2.3.2 Transfert d'apprentissage 17

2.3.3 Coefficient et fonctions de perte 18

2.4 Manipulation de graphe 20
2.5 Conclusion 21
CHAPITRE 3 APPROCHES PROPOSÉES 23
3.1 Ensemble de données 23
3.1.1 Protocole d'annotation avec Fiji 26
3.1.2 Protocole d'annotation avec Ilastik 27
3.2 Approche de segmentation par seuillage 29
3.2.1 Débruitage et restauration de volume 29
3.2.2 Méthode de seuillage 31
3.2.3 Classification de l'avant-plan 33
3.3 U-Net 33
3.3.1 Préparation des données 34
3.3.2 Paramètres d'architecture 36
3.3.3 Hyperparamètres pour l'entraînement 38
3.3.4 Transfert d'apprentissage 40
3.4 Post-traitement des segmentations 41
3.4.1 Reconstruction des parcelles 41
3.4.2 Exploration des cytonèmes 42
3.5 Conclusion 45
CHAPITRE 4 RÉSULTATS EXPÉRIMENTAUX ET DISCUSSIONS 47
4.1 Optimisation de paramètres et d'hyperparamètres 47
4.1.1 Optimisation de la méthode par seuillage 48
4.1.2 Optimisation des paramètres U-Net 53

4.2 Comparaison des méthodes de segmentation 59
4.2.1 Avantages et limitations 61
4.2.2 Améliorations et pistes de recherche 66
4.3 Comparaison de la quantification des métriques 67
4.3.1 Avantages et Limitations 68
4.3.2 Améliorations et pistes de recherche 70
4.4 Conclusion 70
CONCLUSION 72
ANNEXE A ANNEXE 74
INDEX 84
BIBLIOGRAPHIE 87

TABLE DES FIGURES

Figure 1.1 Visualisation d'un volume avec Fiji 1

Figure 1.2 Protocole de calcul pour la longueur d'un cytonème 3

Figure 1.3 Exemple de classification des ROIs 4

Figure 2.1 Exemple de convolution 7

Figure 2.2 Exemples d'éléments structurants 8

Figure 2.3 Exemple de dilatation 8

Figure 2.4 Exemple d'érosion 9

Figure 2.5 Exemple d'utilisation de *watershed* 10

Figure 2.6 Exemple de squelettisation 10

Figure 2.7 Exemples de segmentation avec seuillage 11

Figure 2.8 Architecture U-Net 2D d'origine 13

Figure 2.9 Architecture U-Net adaptée pour 3D 14

Figure 2.10 Représentations des **cartes de caractéristiques** à différents niveaux dans un CNN 15

Figure 2.11 Terminologie pour l'architecture U-Net 16

Figure 2.12 Formation du squelette d'un tube pour clDice 19

Figure 3.1 Vue d'ensemble de la méthodologie proposée 24

Figure 3.2 Comparaison entre volumes Ctrl et Slik 25

Figure 3.3 Exemple d'annotation de volume 26

Figure 3.4 Exemple de segmentation avec Ilastik 28

Figure 3.5 Vue d'ensemble de l'approche par seuillage 30

Figure 3.6 Exemples de déconvolutions avec Richardson-Lucy 31

Figure 3.7 Fenêtrage de l'axe **L** avec Butterworth 32

Figure 3.8 Exemple de division en parcelles 35

Figure 3.9 Ajustement de l'encodeur **VGG16** pour U-Net 37

Figure 3.10 Ensemble des couches pour un bloc de décodage pour un U-Net 38

Figure 3.11 Squelettisation clDice 40

Figure 3.12 Exemple de réassemblage de parcelles 42

Figure 3.13 Séparation des corps de cellules 44

Figure 3.14 Exemple pour notre algorithme de parcours de graphe 45

Figure 4.1 Optimisation d'**hyperparamètres** 48

Figure 4.2 Optimisation de Richardson-Lucy et NLM 49

Figure 4.3 Optimisation du filtre d'affinage 50

Figure 4.4 Optimisation du fenêtrage Butterworth 51

Figure 4.5 Optimisation du filtre médian et de la classification 52

Figure 4.6 Optimisation des données d'entrées pour U-Net 55

Figure 4.7 Entraînement sans poids de classes 56

Figure 4.8 Optimisation des fonctions de perte et d'activation pour U-Net 57

Figure 4.9 Optimisation du taux d'**extinction de neurone** pour U-Net 58

Figure 4.10 Segmentation des volumes de tests 62

Figure 4.11 Segmentation par seuillage 63

Figure 4.12 Segmentation Ilastik 64

Figure 4.13 Segmentation U-Net 2D 65

Figure 4.14 Visualisation des cytonèmes calculés 68

Figure 4.15 Possibilités de croisement de cytonèmes 69

Figure A.1 Classification des régions sur la surface d'une cellule avec l'outil u-shaped3D. 74

Figure A.2 Segmentation par méthode de seuillage globale 76

Figure A.3 Bloc milieu pour notre U-Net sans **encodeur pré-entraîné VGG16** 77

Figure A.4 Bloc d'encodage pour notre U-Net sans **encodeur pré-entraîné VGG16** 77

Figure A.5 Optimisation de la méthode de seuillage globale 78

Figure A.6 Exemples avec Volumentations 79

LISTE DES TABLEAUX

Table 3.1 Prétraitement des données 3D 34

Table 3.2 Paramètres de l'architecture 36

Table 3.3 Hyperparamètres pour l'entraînement 39

Table 4.1 Paramètres par défaut pour l'approche par seuillage 48

Table 4.2 Paramètres optimaux pour l'approche par seuillage 53

Table 4.3 Paramètres et **hyperparamètres** par défaut pour U-Net 54

Table 4.4 Paramètres et **hyperparamètres** optimaux pour U-Net 60

Table 4.5 Comparaison entre nos modèles de segmentation proposés 61

Table A.1 Dimensions des volumes annotés. 75

ACRONYMES

UQAM Université du Québec à Montréal.

IRIC *Institute for Research in Immunology and Cancer*.

UdeM Université de Montréal.

LINUM Laboratoire d'Imagerie Numérique, Neurophotonique et Microscopie.

2D deux dimensions.

3D trois dimensions.

CNN réseau de neurones convolutif.

ISBI *International Symposium on Biomedical Imaging*.

VGG *Visual Geometry Group*.

ViT *Vision Transformer*.

DFS parcours en profondeur.

SVM machine à vecteurs de support.

GAN réseau antagoniste génératif.

ROI région(s) d'intérêt.

RGB rouge, vert et bleu.

ReLU unité linéaire rectifiée.

GELU *Gaussian Error Linear Unit Activation*.

clDice ligne médiane Dice.

IoU *Intersection over Union*.

API interface de programmation d'applications.

W&B Weights & Biases.

Ctrl *Control*.

Slik *Ste20-like kinase*.

GFP protéines fluorescentes vertes.

GPU carte graphique.

PSF fonction d'étalement de points idéals.

AdaWing *Adaptive Wing*.

clDiceAdaWing ligne médianne Dice *Adaptive Wing*.

OCP *one-cycle policy*.

CSV *comma-separated values*.

NOTATION

Nombres

y une matrice avec les valeurs attendues.

p une matrice avec les valeurs prédites.

Unités

GB un gigaoctet.

μm un micron.

nm un nanomètre.

px un pixel.

vx un voxel.

Axes

L axe de profondeur pour un volume.

H axe de hauteur pour un volume ou une image.

W axe de largeur pour un volume ou une image.

D axe des canaux de couleur pour un volume ou une image.

Axes

I image ou volume numérisé.

I' image ou volume numérisé suite à un filtre.

V volume numérisé.

H un noyau pour appliquer un filtre ou un élément structurant pour une opération morphologique.

RÉSUMÉ

Cette recherche est une contribution à un projet cherchant à trouver les mécanismes contrôlant la biogenèse des cytonèmes. Les cytonèmes sont des extensions de cellules spécialisées pour l'envoi de signaux. Ils sont impliqués dans la communication cellulaire lors du développement embryonnaire et dans le développement de tumeur en taille et malignité. Dans le cadre du projet, un microscope confocal est utilisé pour acquérir des volumes de cellules pour l'analyse de leurs cytonèmes à la main en 2D. Afin d'accélérer l'analyse et de mieux prendre en compte le troisième axe des volumes, on cherche à automatiser la quantification de la longueur des cytonèmes dans des volumes confocaux. Le défi principal de cette recherche est qu'aucun travail sur la segmentation de cytonème n'existe à notre connaissance. De plus, la manipulation de données volumétriques impose des limitations matérielles, les volumes à notre disposition contiennent peu de **voxels** de cytonèmes comparativement aux autres ROIs et peu d'entre eux sont annotés.

La quantification des cytonèmes à partir de volumes exige d'abord une segmentation des cytonèmes pour procéder à l'analyse de la morphologie des classes segmentées. Pour nos expériences, un ensemble de données non publiques a été mis sur place. Il consiste en 35 volumes de cellules ayant des cytonèmes, dont 10 sont annotés. Ce mémoire propose 3 approches pour segmenter les cellules. La première approche de segmentation n'utilise pas d'apprentissage machine et les deux autres sont avec l'architecture U-Net en deux dimensions (2D) et trois dimensions (3D). En ordre, elles atteignent un coefficient DiceClDice de 0.7399, 0.7506 et 0.7633 avec nos volumes de tests. Pour chaque méthode, la classification de noyaux cellulaires est problématique puisqu'un noyau est une grande région sombre et il est facilement interprétable comme l'arrière-plan. L'entraînement est restreint à 12 GB de mémoire vive, soit le standard pour une carte graphique. À partir d'un volume où les classes de corps cellulaire et de cytonèmes ont été segmentées, on propose une méthode d'analyse automatisée pour calculer la longueur des cytonèmes associés à chaque cellule.

À moins de changer les méthodes d'acquisition en ajoutant une substance fluorescente aux noyaux cellulaires de sorte à faciliter leur classification, la segmentation du noyau cellulaire nécessite une amélioration afin d'augmenter la fiabilité de notre nouvelle méthode d'extraction de métrique. Dans l'état actuel de ce projet, les métriques de cytonèmes calculées pourront être utilisées pour donner un premier coup d'œil lors de comparaison inter-volume avant de passer à une analyse manuelle.

Mots clés : Segmentation 3D, U-Net, Cytonème, Transfert d'apprentissage.

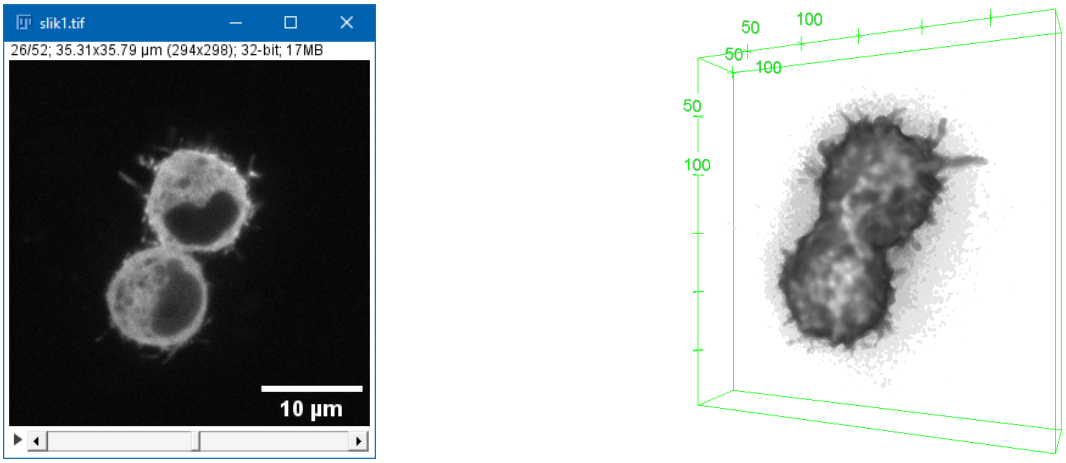
CHAPITRE 1

INTRODUCTION

1.1 Contexte

La segmentation est une tâche de classification de pixels à travers une ou des images. Classifier est la première étape pour pouvoir interpréter et quantifier une image. Avec les avancées des techniques de traitement d'image, il est envisageable d'automatiser non seulement la segmentation, mais aussi la quantification de métriques à partir de celle-ci (Qasim, 2021). L'analyse manuelle d'image permet à un analyste de se revérifier et d'ajuster son approche et ses résultats. Cependant, cette analyse lui coûtera du temps et son interprétation peut s'avérer difficile à reproduire en présence d'incertitudes, menant à des conflits **extra-expert** sur la classification des **voxels**.

Cette dernière problématique est particulièrement présente lors de manipulation de données volumétriques due à la difficulté de visionner un troisième axe en deux dimensions (2D). La représentation d'un troisième axe est typiquement faite à l'aide d'une vue en 2D qui itère sur la profondeur du 3^e axe (Figure1.1a). Une seule image est affichée à la fois, retirant la perception de profondeur du volume.



(a) Vue 2D avec une barre de défilement pour itérer la profondeur du volume.

(b) Vue 3D avec le module *3D Viewer* (Schmid *et al.*, 2010).

Figure 1.1 – Exemples de visualisation d'un volume avec l'outil Fiji (Schindelin *et al.*, 2012).

1.2 Motivation

La caractérisation de la morphologie cellulaire est complexe. L'apparence des cellules est grandement variée, différentes modalités d'acquisitions d'images existent, la complexité logique et computationnelle due à la manipulation de grandes données 3D, etc. À l'exception de composantes de cellules plus recherchées comme le noyau (Caicedo *et al.*, 2019), plusieurs biologistes se basent sur leur interprétation d'images acquises et calculent à la main les métriques **morphologiques** des cellules.

Des experts tentent d'identifier les mécanismes contrôlant la biogenèse des cytonèmes. Ces extensions cellulaires de signalisation sont notamment impliquées dans la communication entre les cellules lors du développement de l'embryon (Zhang et Scholpp, 2019). Surprenamment, il a été démontré que les cytonèmes contribuent au développement de tumeur en taille et malignité (Fereres *et al.*, 2019).

Dans le cadre de notre projet en collaboration avec les experts S. Carreno et B. Rambaud de l'*Institute for Research in Immunology and Cancer* (IRIC) à l'Université de Montréal (UdeM), on cherche à caractériser la morphologie cellulaire suite à une intervention. Dans le cadre de leur recherche, on varie l'expression de gènes potentiellement impliqués dans la régularisation de la biogenèse des cytonèmes de sorte à en influencer leur croissance. Pour visualiser leur expérience, les cellules sont numérisées en 3D à l'aide d'un microscope confocal (Czymmek *et al.*, 1994). L'influence des gènes candidats est estimée en comparant le plus long cytonème moyen par cellules parmi plusieurs conditions.

1.3 Problématique

Afin d'étudier les cytonèmes des cellules observées en 3D, le protocole manuel suivant est utilisé par le IRIC pour calculer la longueur d'un cytonème (Figure1.2) :

1. Sélection du centroïde du corps de la cellule auquel le cytonème appartient dans le volume.
2. Sélection d'une coordonnée 2D pour le début du cytonème.
3. Sélection d'une coordonnée 2D pour la fin du cytonème.
4. À partir du centre du corps de la cellule, placer une sphère C_1 avec un rayon égale à la distance entre le centre et le début du cytonème.
5. À partir du centre du corps de la cellule, placer une sphère C_2 avec un rayon égale à la distance entre

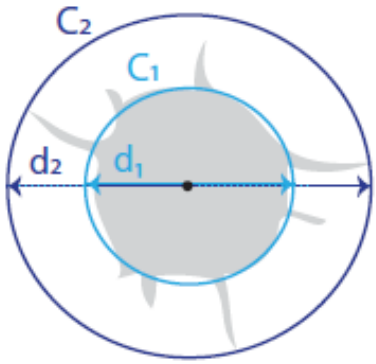


Figure 1.2 – Protocole de calcul pour la longueur d'un cytonème. L'image utilisée est sélectionnée à la main à travers le volume sur l'axe de profondeur. Figure adaptée de notre article (Lemieux *et al.*, 2022).

le centre et la fin du cytonème.

6. Calcul de la différence entre les diamètres d_1 et d_2 divisés par 2.

La sélection des coordonnées est aussi faite à la main en itérant sur l'un des axes du volume, affichant une suite de tranches 2D dans le volume. Comparer la longueur moyenne des cytonèmes par cellule est long et les rayons de cercle à utiliser pour le calcul sont subjectifs. D'autre part, cette méthode ne considère pas la **tortuosité** des cytonèmes, ce qui risque de diminuer la longueur réelle des cytonèmes. Il est important que les métriques calculées soient suffisamment fiables pour publier leur expérience, mais il n'est pas réaliste en termes de temps de faire chaque volume à la main.

On vise à mettre en place un pipeline de segmentation sémantique et de post-traitement de volumes pour accélérer la quantification des résultats d'expérience et avoir une méthode reproductible pour la caractérisation de la morphologie des cytonèmes en 3D. La solution finale doit pouvoir être utilisable par quelqu'un sans expérience en programmation. Pour assurer la vitesse de mise en place d'un pipeline, on propose une méthode de développement itératif pour la segmentation de région(s) d'intérêt (ROI) en 2D ou 3D qui demande peu de ressources computationnelles. Pour l'instant, on arrive à ne pas dépasser une empreinte mémoire de 12 GB sur un carte graphique (GPU) [1] lors de l'entraînement de réseau de neurones convolutif (CNN) (Indolia *et al.*, 2018) 2D ou 3D, mais l'option d'augmenter l'empreinte reste disponible pour un modèle de plus grande envergure.

1. 12 GB est la mémoire standard pour une carte graphique

1.4 Contributions

Dans ce mémoire, on propose 3 approches de segmentation de cellules ainsi qu'un processus de traitement d'image afin d'extraire les métriques utilisées par les chercheurs. Une première passe de traitement d'image de bas niveau est faite pour voir si l'utilisation d'intelligence artificielle est nécessaire. Un article [2] fut rédigé pour la conférence *International Symposium on Biomedical Imaging* (ISBI) 2022 à ce sujet (Lemieux *et al.*, 2022) où il y a été constaté que pour une cellule, les régions de **lamellipode** et la noirceur du noyau cellulaire posent un problème pour une segmentation par seuillage d'intensité global (Figure1.3). Ceci nécessite alors l'utilisation de CNN comme U-Net 2D et/ou 3D et demande une étape de création de sous-volumes afin de réduire la mémoire demandée lors de prédictions.

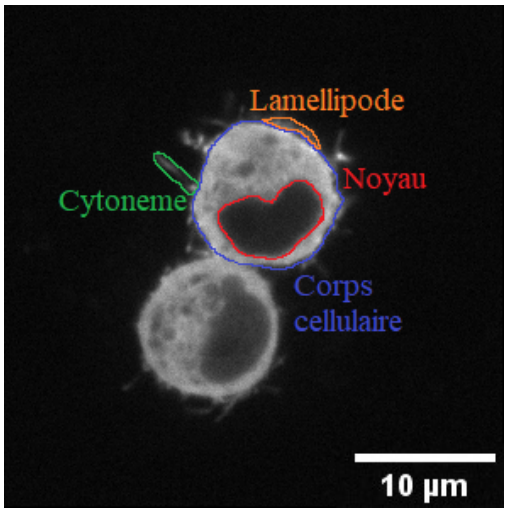


Figure 1.3 – ROIs pour l'image de cellule dans Figure1.1a.

À l'opposé de la segmentation 3D, la segmentation 2D est très accessible, rapide et populaire pour pré-entraîner des modèles de vision par ordinateur. Notamment, le modèle **VGG16** (Simonyan et Zisserman, 2014) entraîné sur ImageNet (Deng *et al.*, 2009) qu'on souhaite utiliser comme encodeur. Pour ces raisons, on propose de commencer à segmenter en 2D. Plusieurs méthodes, dont le transfert des poids 2D appris vers 3D, seront abordées pour accélérer la convergence du modèle. L'ensemble du code est disponible sur le projet GitHub au lien **https://github.com/linum-uqam/carreno-cytonemes**.

D'autre part, le domaine d'intelligence artificielle est primordialement anglais, ce qui pose un problème pour une rédaction française à son sujet. Pour la traduction des termes anglais, on utilise un guide de lexique

2. L'article en question est à la fin de l'annexe section A.

à cet effet (DataFranca, 2022) autant que possible. Les dimensions de volume seront référées comme **LHWD**. De la même façon pour, les dimensions d'une image seront **HWD**.

1.5 Structure du mémoire

Dans cet ouvrage, on se concentre principalement sur la tâche de segmentation sémantique de cellules et ses cytonèmes dans un volume confocal. Pour la structure du mémoire, les prochaines sections seront organisées de la façon suivante :

- Chapitre 2 présente une revue d'articles reliés au domaine de segmentation d'images. On commence en introduisant les méthodes de plus bas niveau ne nécessitant pas d'apprentissage machine. Ensuite, on discute des approches de segmentation plus récentes pour l'entraînement de modèles de segmentation et des U-Nets dans le contexte d'imagerie médicale.
- Chapitre 3 décrit l'annotation d'un ensemble de données biomédicales, les approches de segmentations proposées ainsi que le calcul de métriques morphologiques.
- Chapitre 4 présente nos résultats expérimentaux et compare nos approches entre elles puisqu'il n'existe pas, à notre connaissance, d'antécédents au sujet de la segmentation de cytonèmes.

CHAPITRE 2
MÉTHODES ET CONCEPTS

Ce chapitre porte sur les aspects théoriques liés au projet. On y discute de l'état de l'art pour la segmentation d'image biomédicale présenté dans la littérature. Dans la section 2.1, on discute des convolutions, des opérations morphologiques et de l'analyse d'intensité, qui sont tous des concepts de traitement de bas niveau. Par la suite, la section 2.2 se concentre sur la segmentation d'images basées sur les CNN. On y présente aussi quelques concepts d'optimisation de modèles pour la segmentation sémantique, dont les fonctions de perte. Finalement, la section 2.4 discute de manipulation de graphe pour parcourir les cytonèmes dans nos volumes. En particulier, les opérations de filtre, de morphologie et de segmentation.

2.1 Bases du traitement d'images

Pour le traitement d'images, notre revue de littérature indique qu'il est préférable d'utiliser des réseaux de neurones dont nous discuterons plus en détail dans la section 2.2. Pour utiliser cette approche, un ensemble de données annotées est recommandé s'il en existe un. Autrement, on se tourne vers les méthodes de traitement d'images de bas niveau pour segmenter les ROI d'une **image numérique** I en fonction de critères prédéfinis.

2.1.1 Convolutions

L'opération de convolution vise essentiellement à appliquer un filtre défini sur un petit voisinage sur chaque pixel dans I. L'opération est dénotée $I * H$, où H est un **noyau**[1]. Prenons pour exemple la Figure2.1 où $\{x_0, x_1, ..., x_{14}, x_{15}\}$ est I, $\{w_0, w_1, ..., w_7, w_8\}$ est H et $\{y_0, y_1, y_2, y_3\}$ est I', soit le résultat de la convolution. y_0 est égale à $x_0 \times w_8 + x_1 \times w_7 + ... + x_9 \times w_1 + x_{10} \times w_0$. Soit l'équation suivante où i et j sont des indices d'axes dans un système de coordonnées matricielles :

$$y_0 = \sum_{i=0}^{2} \sum_{j=0}^{2} x[i, j] * w[0 - i, 0 - j] \tag{2.1}$$

1. Par convention, on présume que son point d'origine est au centre

Cette somme est faite sur chaque coordonnée qui peut accommoder les dimensions de H en passant sur chaque colonne de chaque ligne. On note que I' est de plus petites dimensions que I, mais on peut tout de même conserver celles-ci à l'aide de **rembourrage** pour agrandir I avant de convoluer. Les deux types de rembourrage qui seront mentionnés seront le rembourrage réflexif qui miroite l'image et le rembourrage constant qui ajoute des 0.

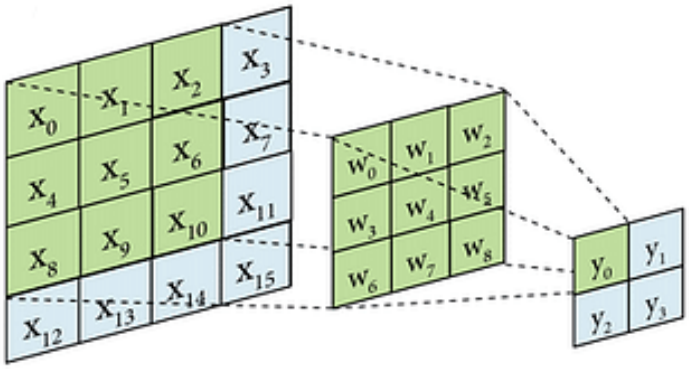


Figure 2.1 – Exemple de convolution où x_i fait partie de I, w_i fait partie de H et y_i fait partie de I'. La figure est adaptée de (Sahoo, 2022).

2.1.2 Morphologie de régions

Pour calculer la longueur et la tortuosité d'un cytonème, il faudra d'abord analyser sa morphologie. La morphologie réfère à la forme d'une région. Plusieurs méthodes de traitement d'images existent pour restaurer et nettoyer la morphologie de régions dans une image numérisée. On se concentre ici sur la morphologie binaire, soit avec les valeurs 0 ou 1. Pour chaque opération morphologique, un élément structurant H est utilisé. En 2D, un H typique est une matrice 3×3 avec une forme de croix ou de carré (Figure2.2). H est déplacé sur chaque coordonnée dans I de la même façon qu'on le ferait pour une opération de convolution. Chaque élément dans H représente une condition booléenne ou une position à ignorer.

L'opération de dilatation est dénotée $I \oplus H$. Chaque pixel dans l'avant-plan de I est remplacé par H. Typiquement, les pixels de **rembourrage** pour I sont considérés comme l'arrière-plan lors de la dilatation. Comme on le voit dans la Figure2.3, I est dilaté avec un H de connectivité 4, ce qui donne I'. Celle-ci est souvent associée à l'expansion d'une région.

L'opération d'érosion est dénotée $I \ominus H$. À l'inverse, elle est associée à la contraction d'une région. Chaque

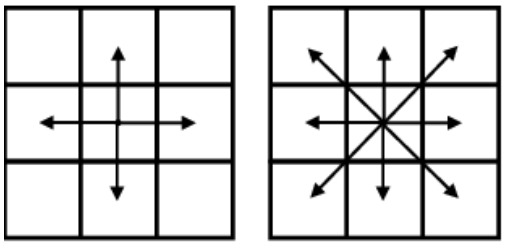
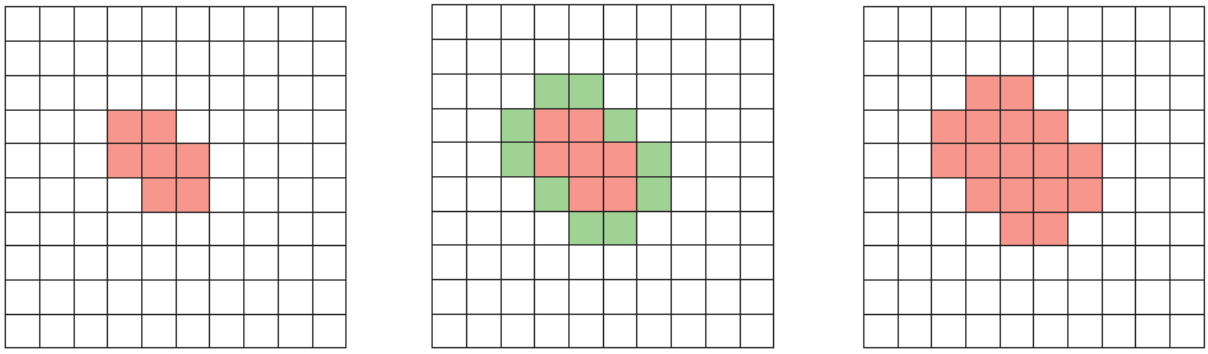


Figure 2.2 – Exemples d'éléments structurants en croix et en carré. La croix peut être interprétée comme ayant une connectivité de 4 et une connectivité de 8 pour le carré. La figure est prise de (Kramm *et al.*, 2021).

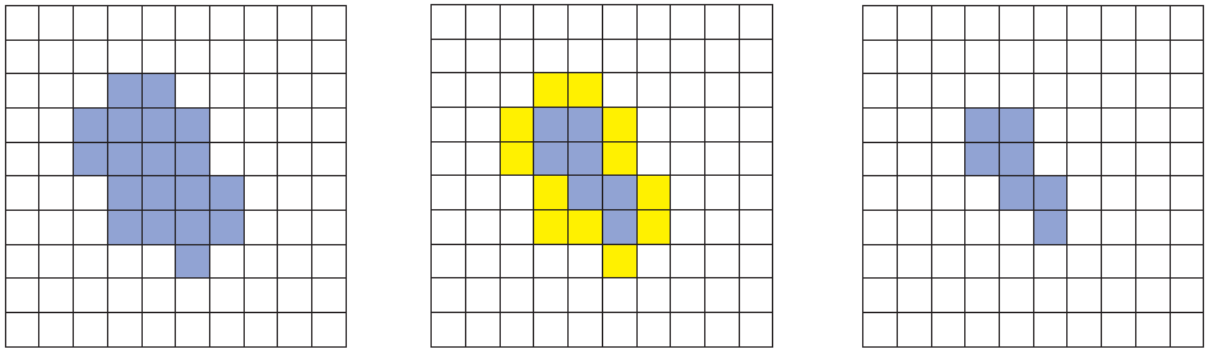


(a) I (b) $I' - I$ en vert (c) $I' = I \oplus H$

Figure 2.3 – Exemple de dilatation avec une connectivité de 4. La figure est adaptée de (Burger et Burge, 2010).

pixel dans l'avant-plan de I est conservée dans I' seulement si elle respecte H. Typiquement, les pixels de **rembourrage** pour I sont considérés comme l'avant-plan lors de l'érosion afin d'éviter de retirer automatiquement les pixels sur les rebords. Comme on le voit dans la Figure2.4, I est érodé avec un H de connectivité 4, ce qui donne I'.

En combinant les opérateurs de dilatation et d'érosion, il est possible de faire les opérations de fermeture et d'ouverture. En commençant par la fermeture, il s'agit de $(I \oplus H) \ominus H$. En appliquant d'abord une dilatation, les régions d'arrière-plan plus petites que H sont remplies. Ensuite, l'érosion servira à inverser l'expansion de l'avant-plan causée par la dilatation. Pour ce qui est de l'ouverture, il s'agit de $(I \ominus H) \oplus H$. En appliquant d'abord une érosion, les régions d'avant-plan plus petites que H sont retirées. Ensuite, la dilatation servira à inverser le rétrécissement de l'avant-plan. Ces deux opérations combinées permettent



(a) I (b) $I - I'$ en jaune (c) $I' = I \ominus H$

Figure 2.4 – Exemple d'érosion avec une connectivité de 4. La figure est adaptée de (Burger et Burge, 2010).

aussi de lisser les contours des régions dans l'avant-plan.

L'opération morphologique « remplissage de trous » a une utilité semblable à l'opération de fermeture, mais fonctionne différemment. Au lieu d'utiliser des érosions ou dilatations, on cherche des régions d'arrière-plan entièrement encerclées par l'avant-plan. En prenant pour exemple Figure2.4b, où les pixels en jaune sont de l'avant-plan et les pixels en bleu sont de l'arrière-plan, ces derniers seront remplis puisqu'ils sont encerclés (Gonzalez et Woods, 2018) (en utilisant un voisinage de 4).

Lors de l'acquisition de volume confocal, il peut y avoir des cellules qui se touchent. Lors de notre analyse, ces cellules risquent d'être segmentées ensemble et il faudra pouvoir distinguer chaque instance. Cette problématique est importante lorsqu'on calcule une métrique comme le nombre moyen de cytonèmes par cellule puisque le nombre total de cellules sera faux lors du calcul [2]. Une méthode populaire pour la séparation d'une région en sous-régions est *watershed* [3] (Neubert et Protzel, 2014). À partir de points d'origines, ces coordonnées sont dilatées jusqu'à ce qu'elles soient arrêtées par un **masque géodésique** ou à la rencontre d'un autre bassin versant. Une carte de profondeur peut être utilisée pour prioriser l'expansion de certaines coordonnées. Cette méthode est efficace pour la séparation de formes circulaires (Figure2.5).

Finalement, l'opération de squelettisation permet d'amincir l'avant-plan jusqu'à ce que chacune de ses régions soit de 1 pixel de large sans les séparer. Par exemple, la figure 2.6 montre un masque binaire de cheval

2. Ne pas séparer les cellules peut doubler ou même tripler le nombre de cytonèmes moyen par cellule

3. La méthode *watershed* se traduit en français comme « ligne de partage des eaux »

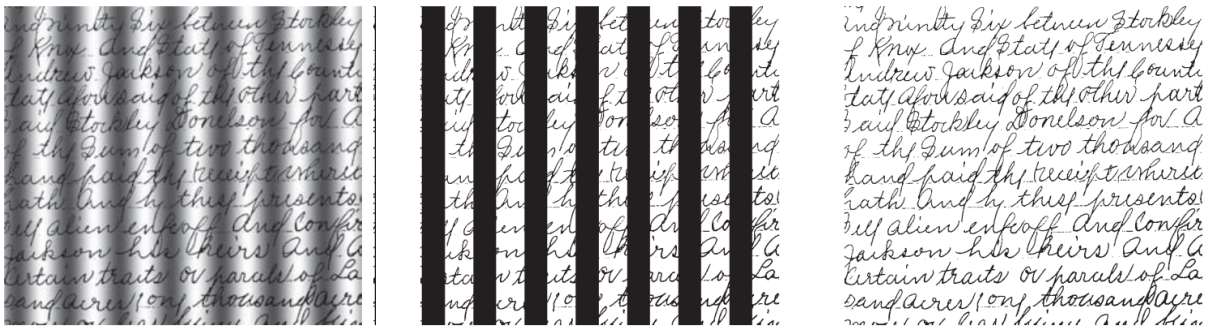
Figure 2.5 – Exemple d'utilisation de *watershed*. La figure est prise de (W. *et al.*, 2023b).

qui est squelettisé. Bien que le squelette soit d'un pixel de large, la connectivité du cheval est conservée. Plusieurs algorithmes sont proposés en 2D pour cette tâche, mais seulement la méthode de Lee Ta-Chih (L. *et al.*, 1994) est implémentée par le module Python scikit-image (W. *et al.*, 2014) en 3D.

Figure 2.6 – Squelettisation d'un masque binaire de cheval. Pris d'un guide pratique pour skimage (W. *et al.*, 2023a)

2.1.3 Segmentation de régions basée sur l'intensité

À partir d'un volume, la segmentation de l'avant-plan est la première étape pour pouvoir identifier les ROIs, soit les cellules. L'une des approches de segmentation est celle de la segmentation des pixels en fonction de leur intensité. En fonction de critères prédéfinis, une valeur de seuillage est calculée à partir de l'histogramme des intensités pour séparer l'arrière et avant-plan. Pour faire ce calcul, 2 familles de méthodes sont proposées. La méthode globale considère l'ensemble des intensités de **voxels** dans l'image pour calculer un seuillage appliqué à l'ensemble de I tandis que la méthode adaptative calcule une valeur de seuillage pour chaque **voxel** en utilisant seulement leur voisinage local. Comme démontré dans la figure 2.7, la méthode adaptative est avantageuse lors de la présence d'une dégradation d'intensité linéaire.



(a) I (b) Seuillage global (c) Seuillage adaptatif

Figure 2.7 – Exemples de segmentation avec seuillage. La figure est prise de (Gonzalez et Woods, 2018).

Avec l'aide de la librairie Python *scikit-image* (W. *et al.*, 2014), on peut facilement visualiser l'algorithme de calcul du seuillage global le plus adapté pour nos données. La fonction *skimage.filters.try_all_threshold* affiche la segmentation d'avant-plan pour les méthodes de seuillage global suivantes :

— Isodata (Ridler et Calvard, 1978) : maximise la distance entre l'intensité moyenne entre chaque classe.
— Li (Li et Lee, 1993) : minimise l'entropie croisée entre les classes.
— moyenne (Glasbey, 1993) : maximise la différence de moyenne entre les classes.
— minimum (Glasbey, 1993) : trouve un maximum local pour chaque classe et conserve le seuil minimum pour les séparer.
— Otsu (Otsu, 1979) : maximise la variance entre chaque classe et minimise la variance intra-classe.
— triangle : sépare l'arrière et avant-plan en fonction de la forme de l'histogramme.
— Yen (Yen *et al.*, 1995) : prend en considération la divergence entre le seuil et I en fonction d'un critère de corrélation maximum.

2.2 Modèle de segmentation

Les approches de traitement d'images présenté ci-haut exigent des paramètres souvent peu flexibles pour différents scénarios. Par exemple, les filtres et les opérations morphologiques sont susceptibles aux changements d'échelles. Pour le cas de cellules, leurs tailles variantes peuvent être interprétées comme des changements d'échelles. Différentes conditions d'acquisition de volume sont présentes dans nos données, ce qui implique des niveaux de bruits inconsistants entre les acquisitions. Les méthodes de vision basées sur l'apprentissage machine permettent d'offrir une segmentation davantage flexible pour supporter ces

variations.

Un article sur la recherche de motifs sur la surface d'une cellule (Driscoll *et al.*, 2019) propose l'utilisation d'une machine à vecteurs de support (SVM) afin de classifier les régions à la surface d'une cellule en utilisant le maillage triangulaire de la surface de la cellule et en partageant l'outil u-shape3D pour vérifier leur approche (FigureA.1). Être capable de classifier une région à la surface d'une cellule comme étant un cytonème serait un bon départ, toutefois, la méthodologie proposée pour obtenir un maillage triangulaire préserve mal les cytonèmes dans nos volumes (FigureA.1a).

On s'intéresse alors à l'état de l'art pour les modèles de segmentation, soit le CNN (Mo *et al.*, 2022) et le **transformeur visuel** ViT (Ruan *et al.*, 2022). Un CNN utilise des couches de convolutions qui contiennent de 1 à plusieurs **noyaux** comme extracteur de caractéristiques, tandis qu'un ViT utilise des **mécanismes autoattentifs** pour identifier les régions pertinentes d'une image en comparant l'ensemble de ses **jetons caractéristiques**. Quoique les caractéristiques globales du **mécasnisme d'attention** puissent surpasser celles des caractéristiques locales d'une convolution, elles sont reconnues pour être plus complexes à entraîner puisqu'elles ne sont pas restreintes à leur voisinage. Bien qu'il existe des recherches traitant sur ce problème (Gani *et al.*, 2022), un ViT nécessite beaucoup de données dues à l'absence de connaissance *a priori* pour combiner ses **jetons caractéristiques** qu'un CNN. Les convolutions déterminent déjà comment combiner les caractéristiques pour le voisinage d'une coordonnée en fonction d'un **noyau**.

Aucun ensemble de données public pour la segmentation de cytonème ne fut trouvé. Mettre en place un ensemble de données suffisamment grand pour entraîner un ViT exigerait beaucoup trop de temps à annoter. Dans le même ordre d'idées, un modèle de segmentation d'instances permettrait l'automatisation de la séparation des instances d'objets qui se côtoient, mais créer un ensemble de données annotées avec des fenêtres qui délimitent les instances de classes n'est pas envisagé pour l'instant dans le cadre de cette recherche. Par ailleurs, les croisements de cytonèmes impliquent qu'un **voxel** peut appartenir à plus d'une instance simultanément.

2.2.1 Architecture d'un U-Net

L'une des approches standard pour la segmentation d'imagerie biomédicale est l'architecture U-Net (Ronneberger *et al.*, 2015). Elle est semblable à celle d'un auto-encodeur. Un auto-encodeur consiste en un encodeur qui compresse le lot de données pris à l'entrée afin d'identifier leurs caractéristiques fondamen-

tales, suivi d'un décodeur qui reconstitue les caractéristiques compressées pour la prédiction. Cependant, un U-Net ajoute des **connexions saute-couches** (*skip connection*) entre chaque niveau de l'encodeur vers le décodeur, permettant d'affiner le décodage ainsi que la segmentation en récupérant certains détails fins perdus lors de la compression (Figure2.8). Pour aider à naviguer dans l'architecture U-Net, voici les mots clés qui seront utilisés :

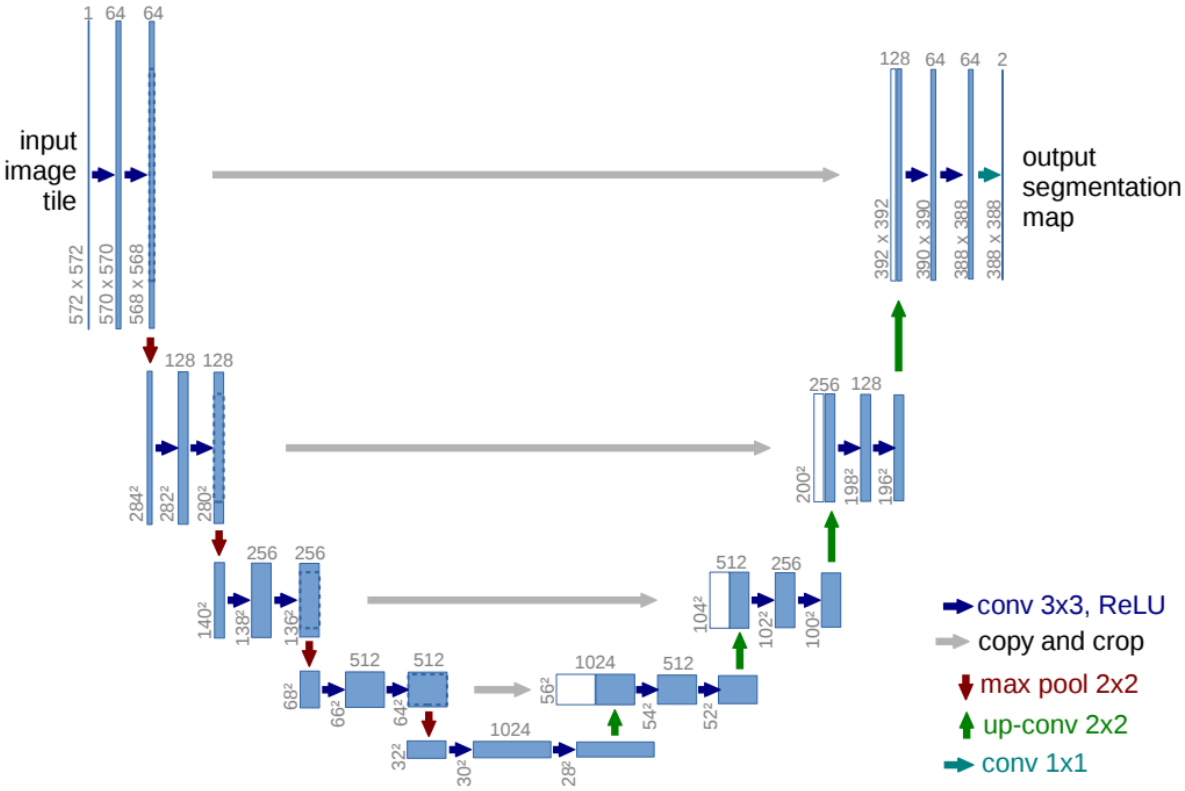


Figure 2.8 – Architecture U-Net 2D d'origine. La figure provient de l'article U-Net (Ronneberger *et al.*, 2015)

— bloc de convolutions : réfère à un ensemble de couches incluant 2 couches de convolutions et excluant toutes les couches d'échantillonnage [4].

— bloc d'encodage : réfère à un bloc de convolutions suivi d'un sous-échantillonnage [5].

— bloc de décodage : réfère à un bloc de convolutions suivi d'un sur-échantillonnage [6].

— bloc milieu : réfère au bloc de convolutions suivant le dernier sous-échantillonnage.

— profondeur : réfère au nombre d'opérations de sous-échantillonnage dans le modèle. Voir la Fi-

4. Traduction de l'anglais pour *pooling* (DataFranca, 2022)

5. Traduction de l'anglais pour *subsampling* (DataFranca, 2022)

6. Traduction de l'anglais pour *resampling* (DataFranca, 2022)

gure2.11.

— niveau : réfère à un bloc d'encodage et de décodage lié par une connexion résiduelle à une certaine hauteur dans le U-Net. Par exemple, le niveau 1 contient le premier bloc d'encodage et le dernier bloc de décodage. Une exception est le bloc du milieu qui réfère à lui-même. Voir la Figure2.11.

Depuis la proposition du modèle U-Net en 2015, des méthodes d'optimisation comme la normalisation de lot (Ioffe et Szegedy, 2015) sont maintenant standards pour faciliter la convergence de modèles [7]. Suite à U-Net 2D, la version 3D (Çiçek *et al.*, 2016) à la Figure2.9 inclut de la normalisation de lot (Ioffe et Szegedy, 2015) avant la fonction d'activation. On y change aussi le nombre de caractéristiques pour la première couche de convolution de chaque bloc de convolutions. Celle-ci est diminuée pour les blocs d'encodage et augmentée pour les blocs de décodage par un facteur de 2. Quoique l'article n'explique pas l'intérêt, ceci devrait diminuer l'impact des couches résiduelles puisqu'elles sont 2 fois plus petites que les cartes d'attributs auxquelles elles sont concaténées. Plusieurs implémentations ignorent ce changement (Müller et Kramer, 2021) (Schmidt, 2023).

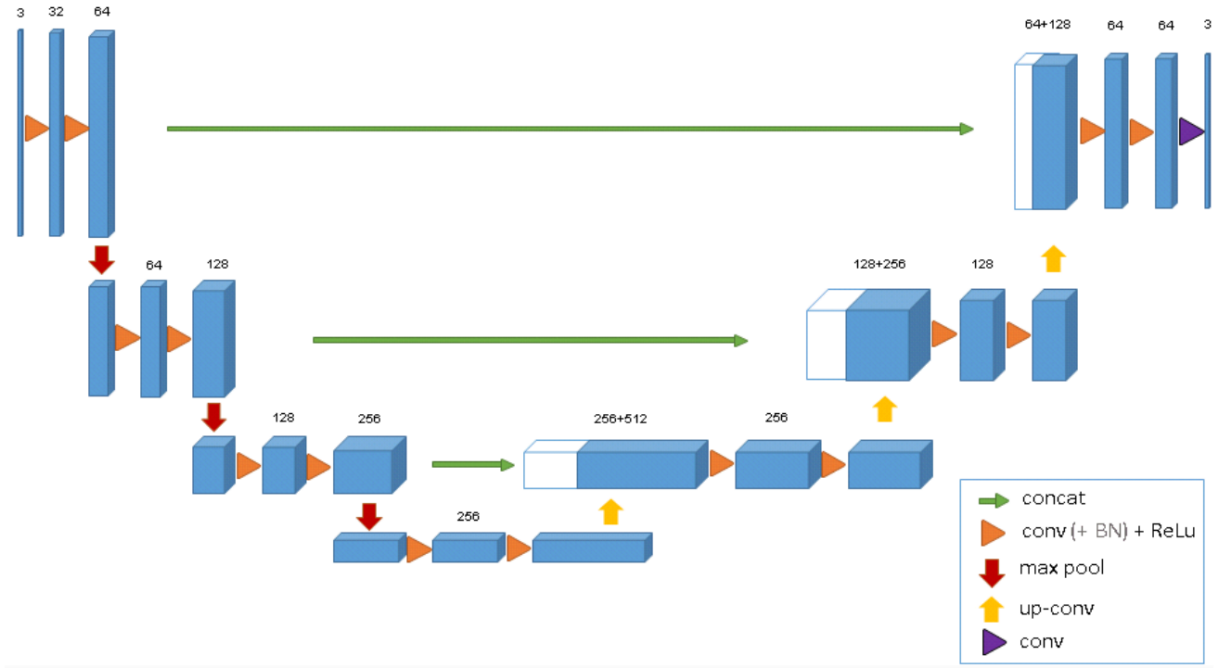


Figure 2.9 – Architecture U-Net adaptée pour 3D. Celle-ci inclut de la normalisation de lot. La figure provient de l'article U-Net 3D (Çiçek *et al.*, 2016)

7. Lorsque la perte de validation se stabilise lors de l'entraînement du modèle

L'utilisation d'un **encodeur pré-entraîné** (*backbone*) pour remplacer l'encodeur d'un U-Net est aussi une option, tant que l'**encodeur pré-entraîné** a une structure compatible, c'est-à-dire qu'elle comporte autant de blocs d'encodage qu'il y a de bloc de décodage dans le U-Net. Dans le cadre de recherche sur la performance de différents **encodeurs pré-entraînés** avec l'ensemble de données ImageNet (Deng *et al.*, 2009), **VGG16** (Simonyan et Zisserman, 2014) réussit étonnamment à rivaliser avec d'autres encodeurs ayant beaucoup plus de paramètres, voire à gagner en précision (Alokasi et Ahmad, 2022)(Zhang *et al.*, 2020). Il est possible de changer l'encodeur d'un U-Net avec un **encodeur pré-entraîné** afin de pouvoir utiliser les poids appris par un modèle pré-entraîné sur ImageNet. Comme montré dans la Figure2.10, les caractéristiques de bas niveau, soit des lignes et coins, peuvent être reprises d'un problème à l'autre. À l'inverse, celles de plus haut niveau devront être mises à jour pour mieux répondre à notre tâche lors de l'entraînement. Le sujet sera élaboré davantage dans la section 2.3.2.

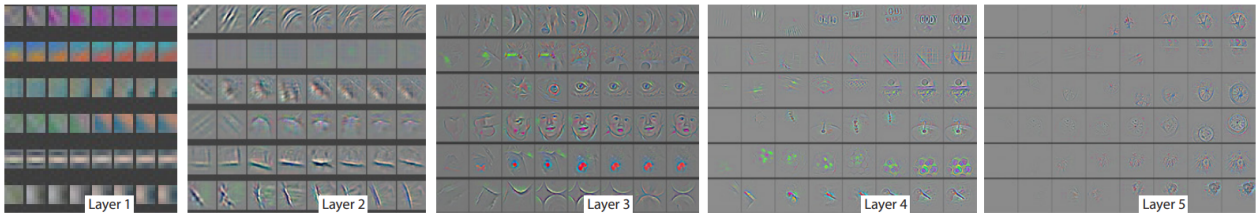


Figure 2.10 – Représentations des **cartes de caractéristiques** à différents niveaux dans un CNN. On y détecte d'abord des lignes droites. Plus les couches de convolution s'ajoutent, plus les lignes se complexifient. La figure provient de l'article U-Net 3D (Zeiler et Fergus, 2013).

L'architecture U-Net peut être adaptée en fonction de la complexité de la segmentation y en modifiant la profondeur et la taille des **cartes de caractéristiques** utilisées par les couches de convolutions. Plusieurs modèles dérivés de U-Net ont été proposés (Siddique *et al.*, 2020), mais ils sont très nombreux et ont tendance à augmenter le coût des calculs sans nécessairement procurer de meilleurs résultats. Par exemple, faire suivre deux U-Net l'un après l'autre (Nisa et Ismail, 2022) ou U-Net++ qui interconnectent les connections de l'encodeur vers le décodeur entre elles avec de nouveaux blocs de convolutions (Zhou *et al.*, 2018). Dans le cadre de cette recherche, la structure utilisée se limitera au format à la Figure2.11.

2.3 Concepts d'optimisation de modèles

La structure d'un U-Net est modulable : l'encodeur peut être remplacé, la profondeur peut être adaptée au besoin, de même pour les couches présentes dans les couches de convolutions, etc. Il n'existe pas de recette absolue et on conseille d'essayer différentes combinaisons de paramètres et d'**hyperparamètres** afin

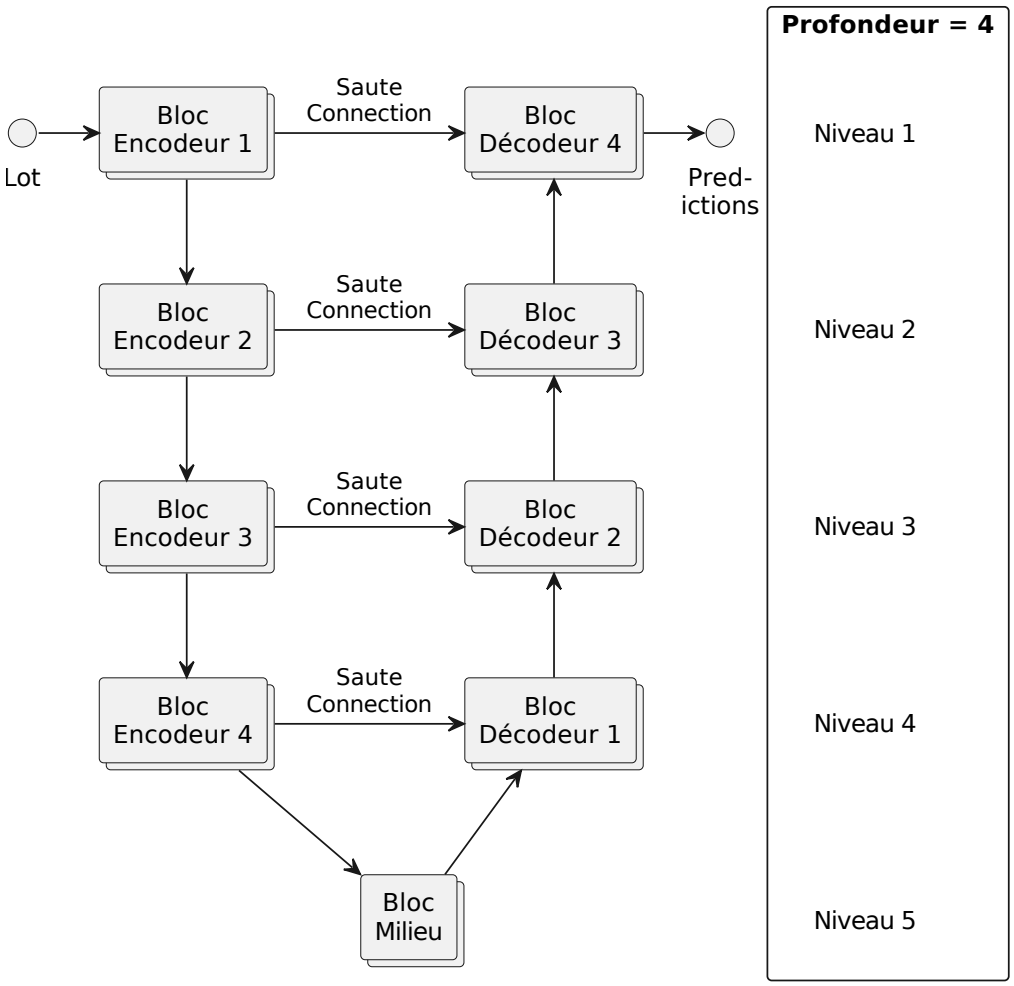


Figure 2.11 – Terminologie pour l'architecture U-Net

d'adapter un modèle à un ensemble de données. Celles-ci seront davantage explorées dans les sections suivantes, mais on s'en tient uniquement à l'apprentissage autosupervisé, le transfert d'apprentissage, SoftSeg (Gros *et al.*, 2020) et les fonctions de pertes.

2.3.1 Apprentissage autosupervisé

L'entraînement supervisé demande d'avoir un ensemble de données où l'on a identifié la classe de chaque ROI. Cependant, nous ne disposons que de très peu de données annotées, d'où l'intérêt de l'auto-apprentissage pour faire un pré-entraînement. En définissant différentes tâches qui ne nécessitent pas d'annotation, on peut potentiellement apprendre des caractéristiques pertinentes pour notre tâche de vision (Newell et Deng, 2020). Une tâche potentielle serait la **reconstruction de lots** où on tente de restaurer une image artificiellement bruitée de sorte à pouvoir utiliser l'image d'origine comme référence. Quelques exemples

de bruitages prouvés efficaces sont de flouter avec des filtres gaussiens (Song *et al.*, 2023), retirer certaines régions avec des **extinctions de neurone** (Pathak *et al.*, 2016a) ou les masques en grille (Chen *et al.*, 2020), on peut entraîner un modèle à bruiter l'image et à utiliser l'image de base comme comparatif avec la prédiction pour la fonction de perte. Cette tâche est souvent faite en utilisant un **réseau antagoniste génératif** (GAN) (Goodfellow *et al.*, 2020), mais celui-ci n'est pas envisageable puisqu'il nécessite beaucoup trop de données pour converger et qu'il est difficile à entraîner.

Il existe tout de même d'autres méthodes qui pourraient s'appliquer à nos données. Par exemple le tri d'images sur la profondeur **L** du volume. Il s'agit d'une méthode typiquement utilisée pour le pré-entraînement de vidéo (Lee *et al.*, 2017) et une vidéo peut être interprétée comme un volume. D'autres méthodes populaires pourraient être explorées dans la continuité du projet (Newell et Deng, 2020).

2.3.2 Transfert d'apprentissage

Un aspect particulièrement utile avec l'utilisation d'**encodeurs pré-entraînés** avec U-Net est l'accès qu'il offre à des encodeurs pré-entraînés sur des ensembles de données comme ImageNet. Bien qu'utiliser ImageNet pour des données biomédicales ne soit pas optimal (Mustafa *et al.*, 2021), on présume que les caractéristiques de bas niveau dans la Figure2.10, telles que les contours et les coins d'objets, seront réutilisées et accéléreront l'apprentissage (Raghu *et al.*, 2019). Cependant, trouver un encodeur pré-entraîné en 3D n'est pas toujours une option. Par exemple, la librairie Python d'apprentissage profond Keras (Chollet *et al.*, 2015) n'offre que des modèles pré-entraînés sur ImageNet en 2D.

Pour transférer les poids de convolution appris en 2D à 3D, on propose de moyenner les cartes de caractéristiques de couches de convolution sur **L**(Yu *et al.*, 2022). Cette méthode donne non seulement accès à l'utilisation d'**encodeur pré-entraîné** 2D en 3D, mais aussi à la réutilisation des poids d'un U-Net 2D dans une version 3D. À l'inverse, cette théorie pourrait aussi être inversée de sorte que les **encodeurs pré-entraînés** sur des images en couleur RGB soient compatibles avec des images en niveau de gris en moyennant les canaux de couleurs **D** appris dans la première couche de convolution sur un nouvel axe **L**. Quoiqu'en pratique, convertir nos données de gris à RGB semble procurer de meilleurs résultats (Hughes, 2022), on pourrait potentiellement simplifier le modèle final en ne prenant pas de canaux de couleurs dupliqués.

2.3.3 Coefficient et fonctions de perte

La fonction de perte utilisée pour l'entraînement joue un rôle majeur dans l'apprentissage du modèle [8]. Lors de l'optimisation de notre modèle, plusieurs d'entre elles devraient être comparées afin de trouver celle qui répond le mieux à nos besoins. L'une des fonctions de perte typique pour la segmentation sémantique est L_{Dice}, décrite dans l'équation suivante :

$$\begin{aligned} Dice &= \frac{2 \times Précision \times Rappel}{Précision + Rappel} \\ &= \frac{2 \times VP}{2 \times VP + VP + FN} \\ L_{Dice} &= 1 - Dice \end{aligned} \tag{2.2}$$

Elle gère bien le déséquilibre de classe, pour des cas comme le nôtre, dans lequel il y a beaucoup plus de **voxels** d'arrière-plan que de **voxels** dans la cellule et les cytonemes.

D'autre part, nnU-Net (Isensee *et al.*, 2018), utilise une fonction de perte composite avec l'entropie croisée (CE) qu'on nomme L_{CeDice}. Celle-ci additionne les pertes de l'entropie croisée et Dice. L'approche de nnU-Net était l'état de l'art en 2018 et demeure un excellent test de référence auquel on peut comparer nos performances. Voici son équation :

$$\begin{aligned} L_{CE} &= -\sum_{c=1}^{M} y_c \times \log(p_c) \\ L_{CeDice} &= L_{CE} + L_{Dice} \end{aligned} \tag{2.3}$$

où M est le nombre de classes, c est une classe, y est la cible et p est la prédiction.

Afin de pouvoir parcourir les cytonèmes pour calculer leur longueur, il est important qu'ils ne soient pas séparés en plusieurs fragments. Pour répondre à cette faiblesse, on se tourne vers 2 approches. La première est la fonction de perte composite avec Dice et ligne médiane Dice (clDice) (Shit *et al.*, 2020). On y calcule la performance de la segmentation sur une sous-partie des classes en amincissants leurs régions sur k itérations (Figure2.12).

En utilisant des **filtres maximums**, on arrive à recréer les opérations de dilatations et d'érosions en niveaux de gris sur chaque couleur de l'axe **D**. À l'aide de ces filtres, on peut estimer le squelette des régions en utilisant l'algorithme 1. Pouvoir quantifier la qualité du squelette permet de l'améliorer et de mieux conserver

8. La mise à jour des poids appris dans notre modèle lors de la descente du gradient

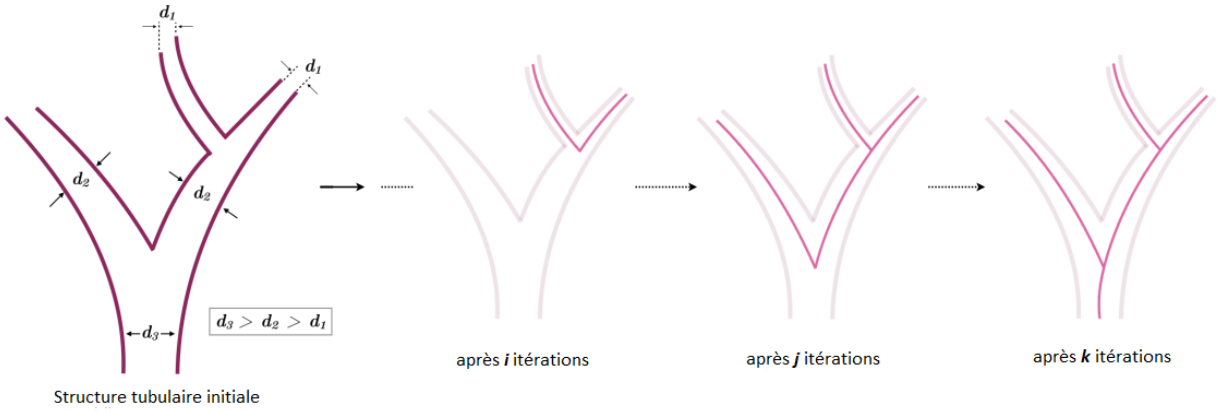


Figure 2.12 – Formation du squelette d'un tube après $i < j < k$ itérations d'amincissement des régions ayant un diamètre d. En mauve est la région à squelettiser et en rose est le squelette obtenu. La figure provient de l'article clDice (Shit *et al.*, 2020).

la connectivité des régions. Lors de l'utilisation de la fonction de perte L_{clDice}, il est conseillé d'y combiner la fonction Dice afin de faciliter la formation de régions autour du squelette avec les équations suivantes :

$$\begin{aligned} Tprec &= \frac{|y \cap p_s|}{|p_s|} \\ Tsens &= \frac{|y_s \cap p|}{|y_s|} \\ clDice &= 2 \times \frac{Tprec \times Tsens}{Tprec + Tsens} \\ L_{clDice} &= 1 - clDice \\ DiceClDice &= (1-\alpha) \times Dice + \alpha \times (1 - clDice) \\ L_{DiceClDice} &= (1-\alpha) \times L_{Dice} + \alpha \times L_{clDice} \end{aligned} \tag{2.4}$$

où y, p sont la cible et la prédiction. y_s et p_s sont le résultat de y et p après un amincissement sur k itérations. $Tprec$ tient pour précision de topologie, $Tsens$ tient pour sensitivité de topologie et α est le ratio entre L_{Dice} et L_{clDice} lors de leur sommation.

Bien que clDice soit une nouvelle fonction de perte, elle démontre d'excellente performance pour la segmentation de classes tubulaires comparativement aux méthodes plus populaires comme Dice, *Intersection over Union* (IoU) et la métrique de précision (Paetzold *et al.*, 2019). Puisqu'un cytonème est tubulaire, celle-ci est bien adaptée pour notre objectif de segmentation.

Algorithme 1 Squelettisation pour clDice

procédure Squelettisation(`Image`, `Itérations`)

`Image'` $\leftarrow$ `Filtre Maximum`(`Filtre Minimum`(`Image`)) ▷ Ouverture

`Squelette` $\leftarrow$ `ReLU`(`Image` $-$ `Image'`)

tant que $i <$ `Itérations` **faire**

`Image` $\leftarrow$ `Filtre Minimum`(`Image`) ▷ Érosion

`Image'` $\leftarrow$ `Filtre Maximum`(`Filtre Minimum`(`Image`))

`Delta` $\leftarrow$ `ReLU`(`Image` $-$ `Image'`)

`Squelette` $\leftarrow$ `Squelette` $+ (1 -$ `Squelette`$) \times$ `Delta`

$i \leftarrow i + 1$

retourne `Squelette`

fin tant que

fin procédure

La deuxième approche pour conserver la connectivité des cytonèmes est SoftSeg, où l'on blâme la fonction de perte Dice et l'activation sigmoïde pour la polarisation excessive des contours de régions [9]. En utilisant une fonction de perte de régression ainsi qu'une activation d'unité linéaire rectifiée (ReLU) normalisée pour la prédiction, on s'attend à obtenir des contours moins polarisés qui pourraient potentiellement mieux conserver la connectivité des cytonèmes. Il faudrait simplement transformer nos **annotations binaires** en **annotations douces**. La fonction de perte régressive proposée est nommée Adaptative Wing ($L_{AdaWing}$) (Wang *et al.*, 2019) et elle utilise l'équation suivante :

$$L_{AdaWing} = \begin{cases} \omega \ln(1 + |\frac{y-p}{\epsilon}|^{\alpha-p}) & \text{if } |y-p| \leq \theta \\ A|y-p| - C, & \text{sinon} \end{cases} \tag{2.5}$$

où y et p reste pareil. α, ω, ϵ et θ sont tous des constantes à virgule flottante à ajuster.

2.4 Manipulation de graphe

Dans le cadre de notre recherche, il faudra pouvoir parcourir un cytonème de son début jusqu'à sa fin pour calculer sa longueur et sa tortuosité. La squelettisation 3D de celui-ci permettrait d'établir un chemin clair à suivre du point A au point B pour le traverser. Si deux cytonèmes se croisent, nous devons pouvoir identifier cette intersection et ne pas dévier du cytonème courant. L'algorithme de parcours de graphe devrait

9. Lorsque les contours sont tracés avec trop de certitude

parcourir un chemin à la fois pour faciliter la gestion des croisements et des branchements de cytonèmes, donc une recherche par profondeur est hors de question. On se concentre alors sur l'algorithme de parcours en profondeur (DFS) (Algorithme 2) pour trouver tous les chemins à travers les squelettes de cytonèmes. Les défis à résoudre avec DFS seront ceux de déterminer les points de départ pour parcourir le squelette des cytonèmes et de savoir comment agir en cas de croisement de cytonèmes afin d'établir des critères de filtrages pour retirer les chemins aberrants. Ceux-ci seront développés dans la section 3.4.

Algorithme 2 Parcours en profondeur

```
procédure DFS(Graphe, Noeud)
    Noeud Visité ← Ajoute(Noeud)
    Pile ← Empile(Noeuds Adjacents(Noeud))
    tant que non Pilevide faire
        Elem ← Top de Pile
        Voisins ← Noeuds Adjacents(Elem)
        pour tout v ∈ Voisins faire
            Continuer ← Faux
            si non v ∈ Noeud Visité alors
                Noeud Visité ← Ajoute(v)
                Pile ← Empile(v)
                Continuer ← Vrai
                arrêter la boucle
            fin si
            si Continuer alors
                Pile ← Dépile
            fin si
        fin pour
    fin tant que
fin procédure
```

2.5 Conclusion

Dans un premier temps, nous avons survolé quelques sujets en liens avec le traitement d'image de bas niveau. Ceux-ci incluent la morphologie mathématique binaire visant à modifier la forme de régions et permettre une meilleure segmentation de régions par histogramme pour identifier les région(s) d'intérêt

(ROI)s. Une revue de la littérature est faite sur les modèles de segmentation afin d'identifier le modèle le mieux adapté à nos besoins et ces optimisations possibles. En ce qui a trait à l'optimisation, on s'en est tenu à identifier de possibles améliorations d'entraînement de modèles avec l'apprentissage autosupervisé pour entraîner sans annotations, le transfert d'apprentissage pour accélérer la convergence du modèle et L_{clDice} ainsi que SoftSeg pour améliorer la segmentation des structures tubulaires. Pour terminer, on a discuté de l'algorithme DFS pour explorer un graphe.

Le chapitre suivant présente les différentes méthodes qu'y ont été élaborées pour la segmentation des cytonèmes : la segmentation basée sur les intensités, la segmentation avec U-Net en 2D et en 3D ainsi que l'optimisation des modèles candidats. On y propose aussi un algorithme de parcours de graphe pour mieux parcourir les cytonèmes segmentés.

CHAPITRE 3
APPROCHES PROPOSÉES

Dans ce chapitre, on décrit les approches discutées pour la segmentation d'image ainsi que le pipeline de post-traitement en charge de l'analyse des cytonèmes. Les différentes méthodes de segmentation utilisées seront celles par seuillage, par U-Net 2D et 3D. Premièrement, la section 3.1 présente les ensembles de données utilisés avec nos expériences. Ceux-ci ne sont pas offerts au public puisqu'elles sont primordialement réservées à un projet de recherche sur les cytonèmes qui n'est pas encore publié. On offre donc une description détaillée de ceux-ci. La section 3.2 traite du processus de segmentation de bas niveau basé sur l'intensité des **voxels** dans le volume. La section 3.3 traite de l'utilisation de modèles U-Net 2D et 3D pour segmenter les cellules dans le volume. La section 3.4 quant à elle propose notre approche pour analyser les cytonèmes segmentés à l'aide d'un algorithme inspiré de DFS qui permet d'explorer le voisinage des **voxels** identifiés comme cytonème.

Une vue d'ensemble est montrée à la figure 3.1 pour nos expériences de segmentation et les liens entrent-elles.

3.1 Ensemble de données

Les volumes utilisés dans le cadre de cette recherche viennent de notre collaborateur B. Rambaud, un étudiant au doctorat en biologie dans le laboratoire de S. Carreno à l'UdeM. Ces volumes ont été acquis avec un microscope confocal Zeiss Observer Z1 équipé d'une caméra Zeiss AxioCam 506 mono. avec un disque tournant par microscopie confocale avec l'utilisation de protéines fluorescentes vertes (GFP) pour faire ressortir les cellules affectées. Pour davantage d'information sur la préparation de cellule, se référer à (Lemieux *et al.*, 2022) section 2.1. Les volumes sont divisés en 2 catégories : *Control* (Ctrl) et *Ste20-like kinase* (Slik). Ctrl contient des cellules avec une expression de gène inchangée tandis que Slik contient des cellules avec une expression de gène modifiée. Visuellement, la différence entre les deux catégories est caractérisée par l'abondance et la longueur des cytonèmes dans les cellules *Ste20-like kinase* (Slik) qui ont tendance à être plus élevée que pour les cellules Ctrl.

Les cellules observées sont sauvegardées sous forme de volumes 3D en niveaux de gris sur 8 bits. Les distances d'axes pour les volumes confocaux sont anisotropiques, c'est-à-dire non uniformes. La résolution

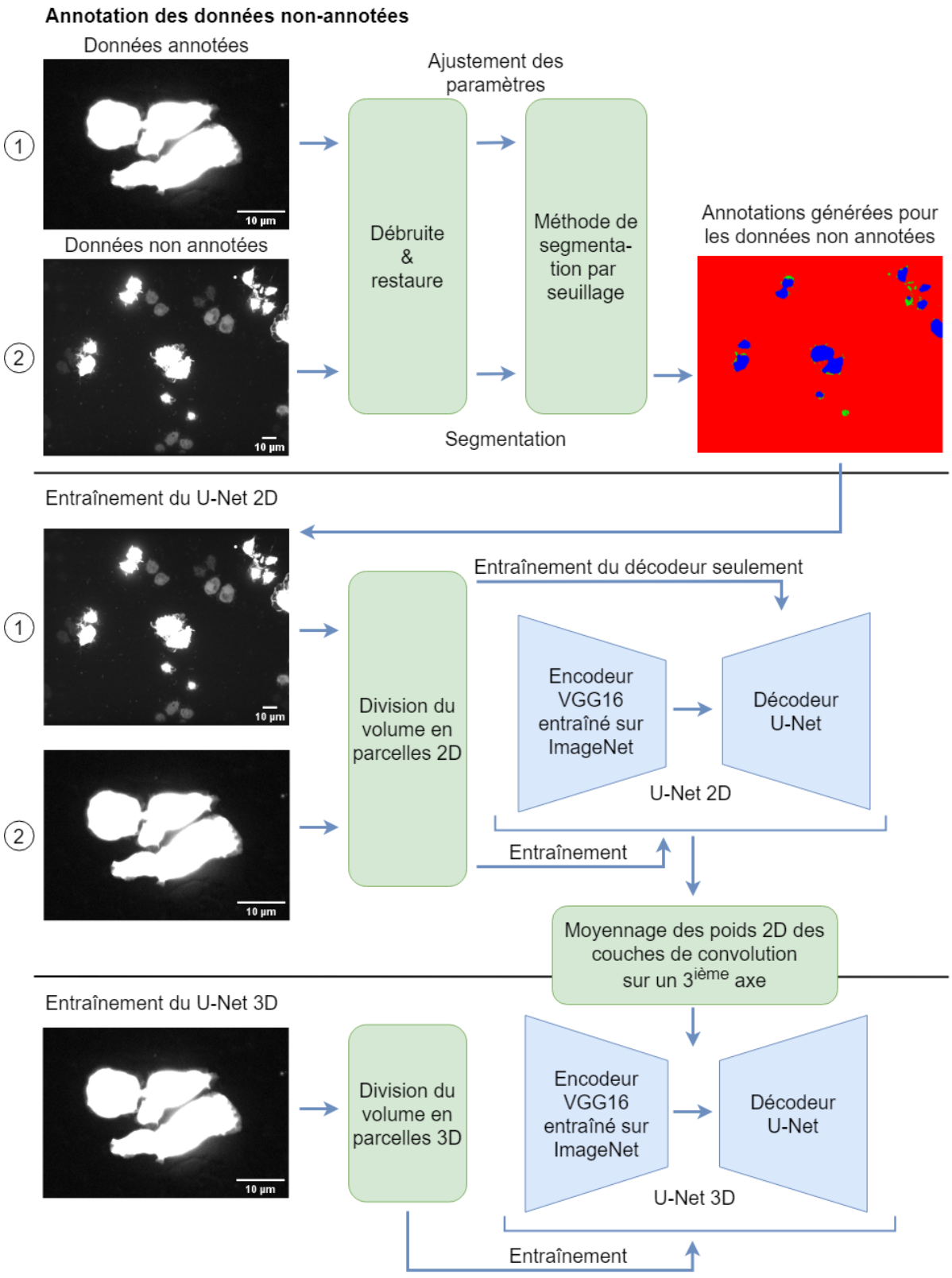
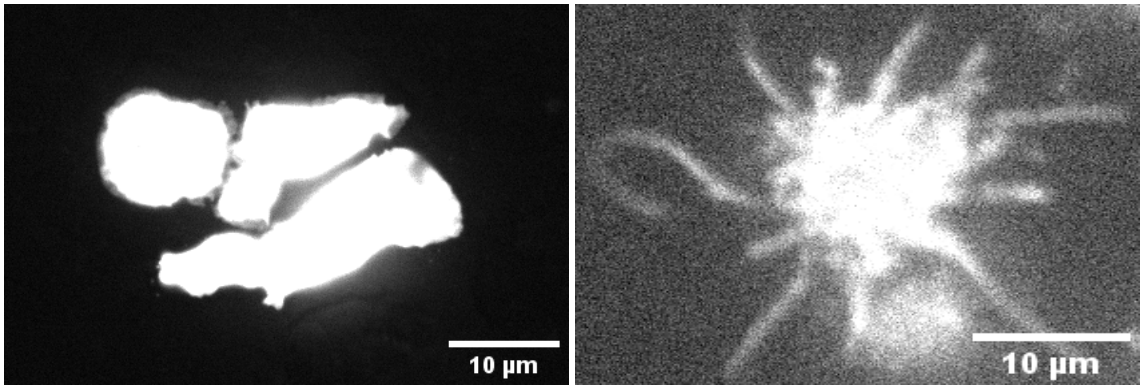


Figure 3.1 – Vue d'ensemble de la méthodologie proposée

latérale est de 120 nm, tandis que la résolution axiale est de 260 nm. Quoique ce ne soit pas toujours le cas, les volumes *Control* (Ctrl) sont majoritairement surexposés [1] et certains volumes Slik ont une présence de bruit sel et poivre [2] et de Poisson [3] élevée comme on le démontre à la figure 3.2.



(a) Volume Ctrl2, tranche 11. (b) Volume Slik3, tranche 10.

Figure 3.2 – Comparaison entre volumes Ctrl et Slik. En général, Slik exprime davantage de cytonèmes en termes d'instances et longueurs.

D'abord, un ensemble de 25 volumes Slik non annotés a été fourni par nos collaborateurs pour de l'apprentissage auto-supervisé. Ils sont tous de dimensions $39 \times 1104 \times 1376 \times 1$ pixel (**LHWD**) en niveau de gris sur des entiers de 8 bits. À notre connaissance, il n'existe pas d'ensemble de données annotées pour une tâche de segmentation des cytonèmes. B. Rambaud a fait l'annotation de cytonèmes et corps de cellules pour 10 volumes avec Fiji et Ilastik (Berg *et al.*, 2019) comme outils. Fiji est un programme Java standard pour le traitement d'images. Dans la section 3.1.1, on décrit comment il est utilisé pour l'annotation de nos données. Ilastik est un regroupement d'outils interactifs pour l'entraînement de forêt aléatoire pour la segmentation. Parmi les volumes annotés, 4 sont Ctrl et 6 sont Slik. Chacun est une sous-section du volume centrée sur un ensemble de cellules exposées à du GFP (FigureA.1). Puisqu'il s'agit de volumes en niveau de gris, la longueur de l'axe **D** est de 1 pour chacun d'entre eux.

Suite au protocole d'annotation avec Fiji, les annotations de corps de cellules (C_o) et cytonèmes (C_y) sont séparées dans différents volumes. Pour pouvoir les utiliser, elles furent combinées dans un volume RGB. En rouge l'arrière-plan, en vert C_y et en bleu C_o (Figure3.3). Puisque B. Rambaud est le seul expert à avoir

1. Présence de saturation des pixels dans l'image
2. Type de bruit où on observe des pixels noirs et blancs aberrants à travers l'image
3. Type de bruit qui dépend de l'intensité du signal lors de l'acquisition.

annoté l'ensemble de données, aucun conflit **extra-expert** n'a lieu. Ceci pose un problème pour l'intégration des annotations douces à nos approches. Pour y remédier, on propose 3 approches. La première est d'appliquer un filtre passe-bas pour flouter les annotations et ainsi simuler artificiellement l'ambiguïté de la classification. La deuxième est d'utiliser l'outil Ilastik pour prédire l'annotation douce des volumes à partir d'une petite fraction de **voxels** classifiés à la main. La dernière est de moyenner les annotations avec la prédiction de classification d'Ilastik de sorte à combiner la précision d'un expert et l'incertitude du modèle entraîné.

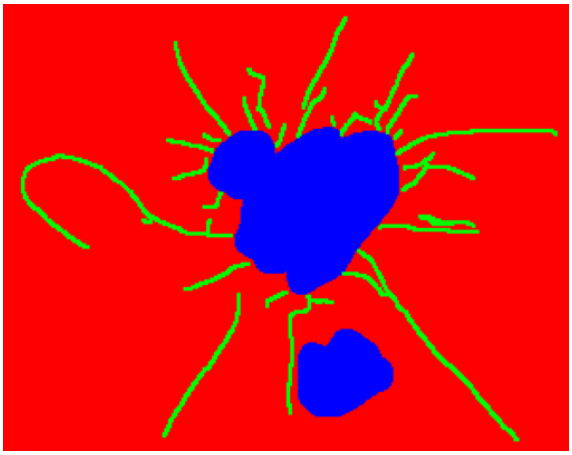


Figure 3.3 – Exemple d'annotation pour nos données. L'arrière-plan est rouge, les cytonèmes sont verts et les corps de cellules sont bleus. L'image provient du volume Slik3.

3.1.1 Protocole d'annotation avec Fiji

Fiji (Schindelin *et al.*, 2012) est l'outil qui a été principalement utilisé pour annoter les volumes, prenant une trentaine de minutes par volume. Il vient avec plusieurs extensions pour le traitement d'images, dont certaines pour l'annotation de **voxels**. Pour ce faire, on met d'abord l'environnement de travail en place en ouvrant un volume comme une pile [4] d'images. On débruite l'image avec la **moyenne non-locale** (NLM) (Coll et Morel, 2011) en utilisant son estimation du sigma ainsi qu'un filtre d'affinage avec un rayon de 50 pixels et un poids de masque de 0.6. On convertit le type de l'image en format RGB pour colorer les différentes classes.

Dans notre cas, on utilise les couleurs bleues et rouge pour l'annotation des cytonèmes et des corps cellulaires. Les cytonèmes nécessitent des détails fins, donc on utilise l'outil *pencil tool* à 2 pixels de diamètre,

4. Traduit de *stack*

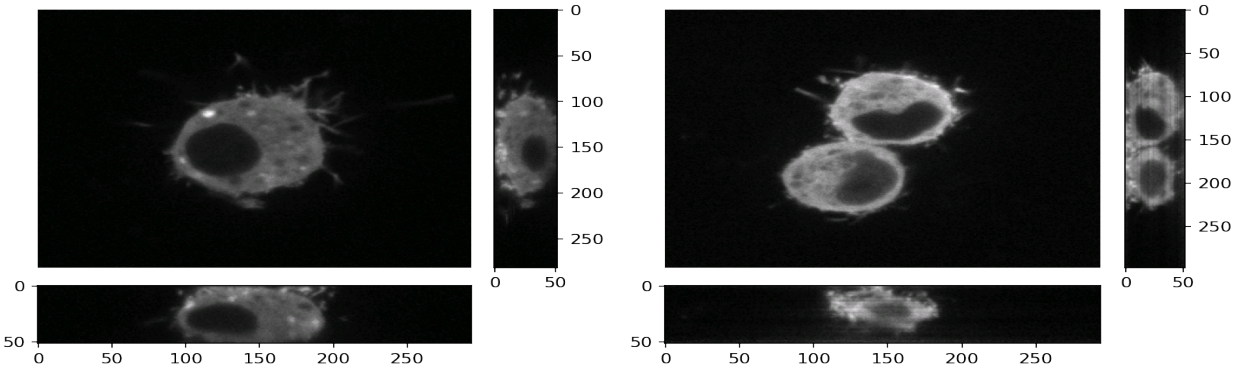
tandis que les corps cellulaires sont dessinés avec l'outil *paintbrush tool* à taille variée. Pour obtenir une annotation binaire, on applique un seuillage à 0 sur chaque couleur sur l'axe **D**. Puisqu'on annote sur 2 couleurs, le résultat de la segmentation donne 2 masques binaires, soit un par classe.

3.1.2 Protocole d'annotation avec Ilastik

L'outil Ilastik (Berg *et al.*, 2019) fut découvert plus tard durant notre projet et fut peu utilisé. Son attrait principal est son excellente interactivité qui permet de visionner à l'instant la prédiction du modèle suite à un bref entraînement avec les annotations dessinées. La procédure d'annotation qui suit prend moins de cinq minutes par volume. Dans un projet Ilastik de classification de cellules, on utilise un seul volume partiellement annoté par modèle de forêt aléatoire pour l'entraînement. D'abord, on classifie partiellement chacune des classes sans trop de détails pour faire une première passe. Ensuite, en fonction de la segmentation obtenue à partir de l'apprentissage développé jusqu'à présent, on ajuste les annotations pour l'entraînement avec davantage de classification manuelle.

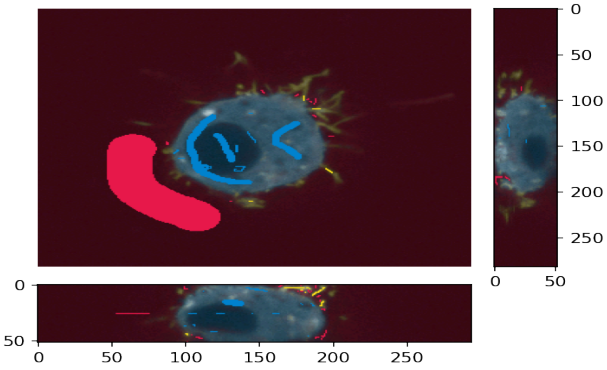
Une observation faite est qu'un modèle n'arrive pas à généraliser la segmentation apprise à plusieurs volumes. On prend pour exemple V^1 (Figure3.4a) et V^2 (Figure3.4b) pour l'entraînement d'un modèle en utilisant l'ensemble des paramètres proposés avec un projet de classification de pixels comme conseillé par l'un des développeurs d'Ilastik (Dominik Kutra, 2022). V^1 et V^2 sont des volumes confocaux Slik de l'ensemble de données annotés par B. Rambaud. Dans Figure3.4c, on annote partiellement V^1 (traits opaques) jusqu'à ce que la prédiction (teinte de couleur) soit satisfaite pour une bonne segmentation. Le modèle n'arrive pas à bien segmenter l'intérieur et l'extérieur du corps de cellule pour V^2 (Figure3.4d). Pour y remédier, on annote V^2 de la même façon que V^1 (Figure3.4e) où la prédiction est basée sur A^1 et A^2. Après avoir corrigé le modèle pour mieux segmenter A^2, on n'arrive plus à bien segmenter le corps de cellule pour A^1 (Figure3.4f).

Peu importe le nombre de corrections apportées, le modèle n'arrivera pas à généraliser de façon satisfaisante son apprentissage pour plus d'un volume (Figure3.4). On conclut qu'il ne s'agirait pas d'un bon modèle de segmentation avec nos volumes. Cependant, en surajustant le modèle pour un seul volume, on obtient une segmentation décente pour le temps investi à l'entraîner.

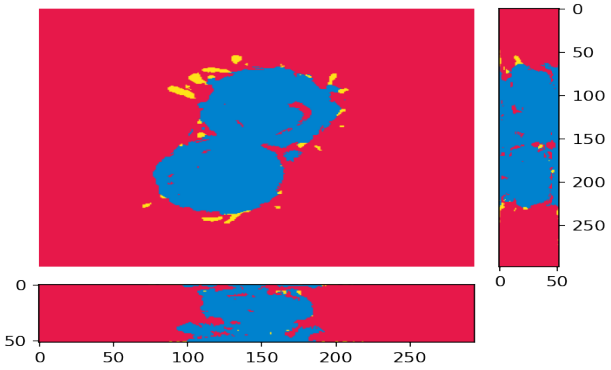


(a) Slik volume (V^1)

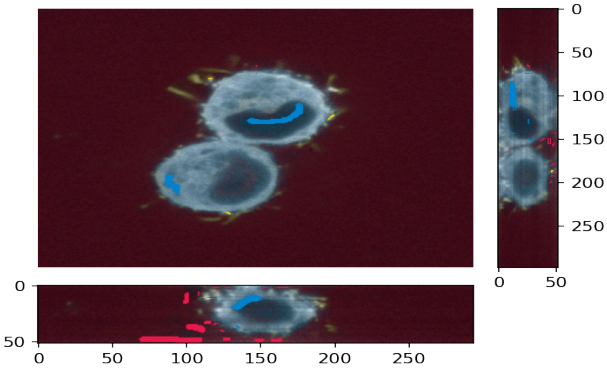
(b) Slik volume (V^2)



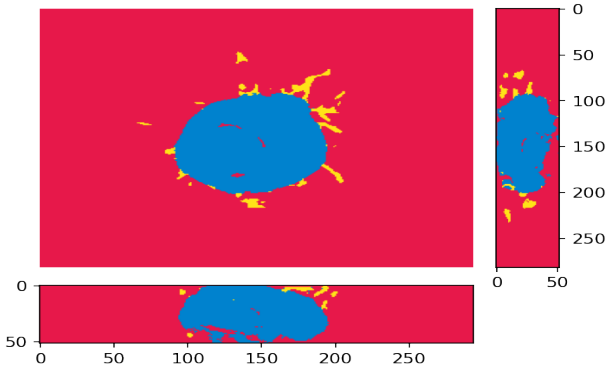
(c) Annotation partielle de V^1 (A^1)



(d) Segmentation de V^2 avec A^1



(e) Annotation partielle de V^2 (A^2)



(f) Segmentation de V^1 avec $A^{1,2}$

Figure 3.4 – Scénario d'entraînement de forêts aléatoires avec plus d'un volume. Chaque image est une tranche du volume confocal pour l'indice à la moitié de la longueur de son axe correspondant (rouge : arrière-plan, bleu : corps de cellules, jaune : cytonèmes).

3.2 Approche de segmentation par seuillage

L'annotation d'ensembles de données est une longue tâche. Particulière en 3D étant donné que la majorité des outils d'annotation présentent un volume comme une série d'images, retirant l'aspect de profondeur. Dans l'espoir d'éviter d'entamer cette étape, il est recommandé de vérifier si les données peuvent d'abord être traitées avec une technique de seuillage. Représentons une image avec l'équation suivante :

$$I(x,y)' = I(x,y)G(x,y) + B(x,y) \tag{3.1}$$

Où $I(x,y)'$ est l'ensemble des pixels pour l'image, $I(x,y)$ est une version parfaite de $I(x,y)'$, $G(x,y)$ est la dégradation linéaire [5] et $B(x,y)$ est le bruit. Puisqu'un seuillage global performe très mal en présence de $G(x,y)$ locale, $I(x,y)'$ doit être restauré avec un filtre passe-haut avant d'effectuer le seuillage de sorte à réduire l'influence de $G(x,y)$. Le dilemme est qu'un filtre passe-haut augmente aussi la quantité de $B(x,y)$.

La figure 3.5 montre les grandes lignes de notre approche.

3.2.1 Débruitage et restauration de volume

L'objectif est d'obtenir une bonne approximation de $I(x,y)$ à partir de $I(x,y)'$ avec un pré-traitement des données. En se basant sur les méthodes de pré-traitements décrites dans un article sur la classification de région à la surface d'une cellule en 3D (Driscoll *et al.*, 2019) et notre publication précédente sur ce sujet dans l'annexe (Lemieux *et al.*, 2022), on y établit un nouveau processus de pré-traitement afin d'atténuer $G(x,y)$ et $B(x,y)$ dans la première section de Figure3.5.

Avec la fonction d'étalement de points idéals (PSF) expérimentale du microscope confocal, on utilise l'algorithme de déconvolution Richardson-Lucy (Richardson, 1972) sur 25 itérations pour réduire la dégradation introduite par le processus d'imagerie (Figure3.6). La PSF est mesurée à l'aide de billes fluorescentes (Haeberlé *et al.*, 2001) pour obtenir une déconvolution plus juste. On propose optionnellement d'utiliser NLM pour tenter d'améliorer l'uniformité des ROIs. Ensuite, les contours de cellules sont rehaussés avec un filtre d'affinage avec un rayon de 6.0 μm. Le résultat du filtre est la somme du volume avec le filtre multiplié par un facteur de 8. Finalement, pour combattre $B(x,y)$ de type sel et poivre dans les volumes Slik, on utilise

5. Typiquement lié l'illumination non uniforme des **voxels** lors de l'acquisition

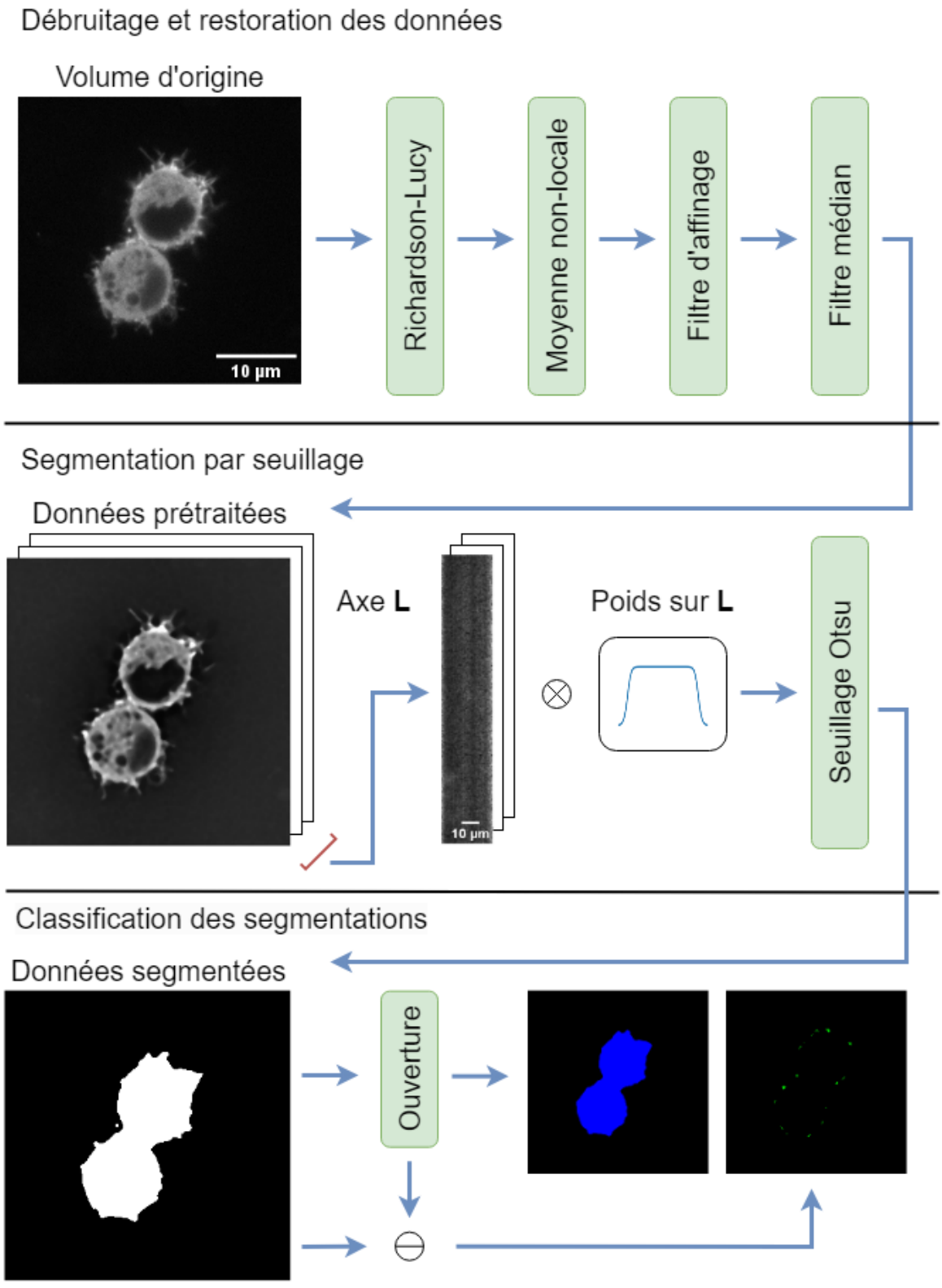


Figure 3.5 – Vue d'ensemble de l'approche par seuillage.

un filtre médian avec des côtés de 600 nm, soit $3 \times 5 \times 5$ vxs.

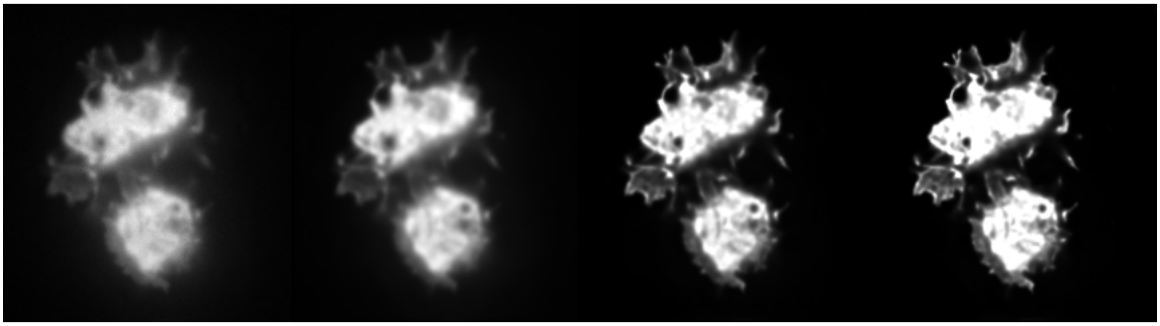


Figure 3.6 – Exemples de déconvolutions avec Richardson-Lucy avec Ctrl1 pour 0, 5, 25 et 50 itérations.

3.2.2 Méthode de seuillage

On souhaite classifier les classes cytonèmes et les corps de cellules pour calculer les métriques suivantes : le nombre moyen d'instances de cytonèmes par cellule et leur longueur. Pour ce faire, il faut d'abord identifier les **voxels** auxquels les cellules appartiennent. On observe que le niveau d'illumination et de clarté varie en fonction de l'indice de l'axe **L**. $G(x, y)$ est plus élevé sur les indices éloignés du centre de l'axe parce que l'effet de l'atténuation du signal avec la profondeur et l'ajustement (probable) du gain électronique sur le détecteur du microscope pour compenser. Cette atténuation donne un effet similaire à une image hors focus.

Suite à nos observations en essayant les méthodes de segmentation proposées dans la section 2.1.3, le seuillage de Li obtient des cytonèmes clairement définis sans exagérer les dimensions des corps de cellules (FigureA.5). Les méthodes de seuillage local Niblack (Niblack, 1986) et Sauvola (Sauvola et Pietikäinen, 2000) sont aussi envisageables pour combattre $G(x, y)$, mais l'arrière-plan de nos volumes est si disproportionnellement grand qu'à moins d'utiliser des régions locales très grandes, on segmente beaucoup de faux positifs, car les régions locales parcourues n'ont souvent aucune présence de cellules.

Pour atténuer $G(x, y)$ avant la segmentation, on propose de faire une opération de fenêtrage sur l'axe **L** afin d'amoindrir les indices inférieurs et supérieurs sur **L** et ainsi améliorer la sélection du paramètre pour le seuillage. Pour ce faire, on multiplie une distribution Butterworth sur l'axe **L** (Figure3.7). La distribution

utilise l'équation suivante :

$$H(x) = \frac{1}{1 + [D(x-a)/b]^{2c}} \tag{3.2}$$

Où $D(x)$ représente la distance entre chaque point (x) et a, b, c sont des valeurs paramétrables. Dans notre cas, pour l étant le nombre de valeurs x, $a = 0.5l$, $b = 0.95l$ et $c = 0.09l$ afin de pouvoir gérer différente longueur d'axe **L**. Pour un volume $50 \times 200 \times 200$ vxs, on discrétise (x) sur 50 valeurs entre la longueur le l'axe $-\mathbf{L}/2$ à $\mathbf{L}/2$.

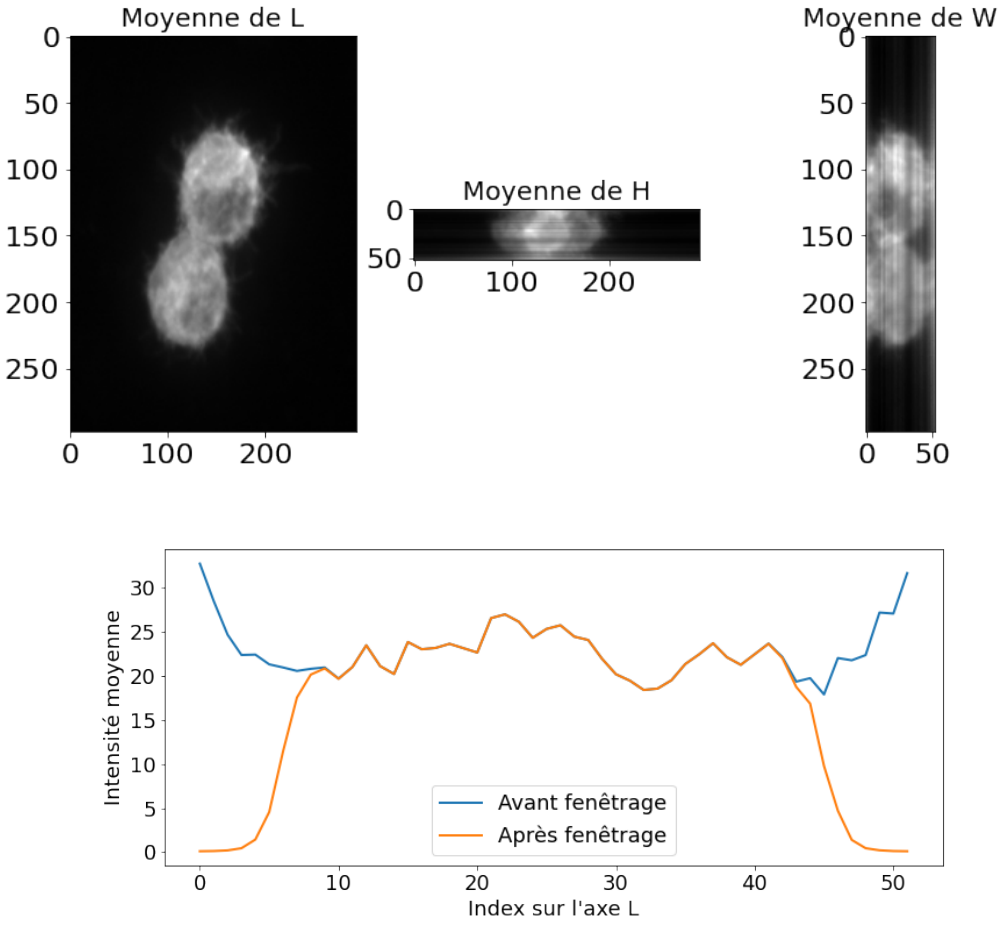


Figure 3.7 – Fenêtrage de l'axe **L** avec Butterworth pour Slik1.

3.2.3 Classification de l'avant-plan

Une fois que la cellule est segmentée, les **voxels** de l'avant-plan doivent être séparés entre 2 classes : un corps de cellule et un cytonème. Les cytonèmes sont beaucoup plus minces que les corps de cellules. Comme démontré dans la dernière étape dans la figure 3.5, on propose d'utiliser une ouverture morphologique binaire pour distinguer les deux classes. Les cytonèmes sont perdus lors de l'érosion avec un **élément structurant** sphérique avec un rayon de 260 nm et le résultat de la dilatation sera interprété comme étant des corps de cellules. La différence entre la cellule et son corps sera interprétée comme étant des cytonèmes.

On s'attend tout de même à mal segmenter un corps de cellule puisqu'il contient un noyau de cellule sombre comme l'arrière-plan. On propose alors de faire une opération de fermeture binaire avec quelques étapes intermédiaires pour les remplir. D'abord, chaque corps de cellule est isolé et dilaté avec un **élément structurant** sphérique ayant un rayon de 1000 nm. Ceci est pour aider à ne pas laisser de trou à la surface de la cellule sans combiner des corps de cellules qui étaient déjà séparés. Ensuite, on applique une opération de remplissage de trou au cas où la dilatation n'aurait pas été suffisante pour remplir le noyau de cellule. Finalement, une érosion est faite avec le même **élément structurant** pour compléter l'opération de fermeture sur les corps de cellules.

Par ailleurs, si le **lamellipode** partage les mêmes niveaux d'intensité que les cytonèmes, il risque d'être mal classifié. On se tourne alors vers l'utilisation de CNN pour interpréter les formes et contours des **voxels** dans la section suivante.

3.3 U-Net

Lorsqu'il s'agit d'imagerie médicale, U-Net est l'architecture classique pour les problèmes de segmentation. Comparativement à notre première approche, un CNN n'est pas limité à l'intensité des **voxels** pour identifier les **voxels** de cellule et n'aura pas besoin de post-traitement pour faire la distinction entre les cytonèmes et corps cellulaires. Son implémentation suit en grande partie la version proposée par l'auteur 0. Ronneberger. Des couches de normalisation de lot et un **encodeur pré-entraîné VGG16** ont été ajoutés pour accélérer la convergence du modèle. Quoique nos données soient des volumes, faire une première passe en 2D est moins complexe et plus accessible en termes de matériel informatique qu'en 3D.

Nous verrons que la majorité des expériences 2D peuvent être récupérées en 3D. De plus, obtenir de bons résultats à ce stade avec un modèle 2D pourrait indiquer que l'aspect de profondeur n'est pas significatif pour la segmentation d'une cellule. Quoique dans notre cas, la précision des cytonèmes doit être très précise. Sans la profondeur du volume, un cytonème peut paraître détaché de lui-même sur une tranche du volume.

3.3.1 Préparation des données

Le tableau 3.1 contient la préparation faite pour les données d'entraînement des modèles.

Table 3.1 – Prétraitement des données 3D. Pour les poids, C_y est la classe des cytonèmes, C_o est la classe des corps de cellules et $\overline{C_o \cup C_y}$ est la classe d'arrière-plan. Pour les annotations douces, σ est pour le filtre gaussien appliqué sur les axes **LWH**.

Configurations		**U-Net 2D**	**U-Net 3D**
Prétraitement	**Échelle**	Standardisée	
	Sous-volume	$1 \times 192 \times 192 \times 3$	$16 \times 80 \times 80 \times 3$
	Poids	$C_y = 38.54\,; C_o = 4\,; \overline{C_o \cup C_y} = 0.37$	
	Annotation	Douce (**L**$\sigma = 0.692\,$; **HW**$\sigma = 1.5$)	

Il est impossible de mettre un volume en entier dans un CNN avec les ressources computationnelles à notre disposition due à l'ampleur de ceux-ci. En général, les dimensions de nos volumes annotés sont autour de $40 \times 300 \times 300\ px$. Avec la stratégie de diviser pour régner, on peut séparer un volume en parcelle pour diminuer les dimensions des entrées dans un CNN. Planifiant d'utiliser un U-Net, il faut respecter que la longueur d'un axe dans nos sous-volumes est :

$$A_l = \mathbb{Z}^+ H_l^d \tag{3.3}$$

où A_l est la longueur de l'axe du volume, d est le nombre de couches de **sous-échantillonnage par valeur maximale** dans l'encodeur du U-Net et H_l est la dimension d'axe des filtres de sous-échantillonage utilisés. Si on ne respecte pas cette contrainte, les couches de **sous-échantillonnage par valeur maximale** ne permettront pas de diviser par 2 les longueurs d'axes. Dans notre cas, $H_l = 2$ et $d = 4$. La grandeur minimale de

A_l est 16. On observe qu'à l'exception de l'axe **L**, maximiser leur longueur est bénéfique à la segmentation. Cependant, ceux-ci sont limités par notre mémoire GPU à 12 GB.

Lors de l'entraînement, les parcelles sont prises aléatoirement à travers le volume. Cependant, pour segmenter un volume entier, on propose de prendre une parcelle de taille égale à celle de l'entrée du modèle et d'en prendre une nouvelle à chaque intervalle de la moitié de la taille de parcelle. Il peut être nécessaire d'ajouter du **rembourrage** pour respecter l'intervalle de sélection (Figure3.8). On conserve l'ordre des parcelles de sorte à les rassembler dans la section 3.4.1.

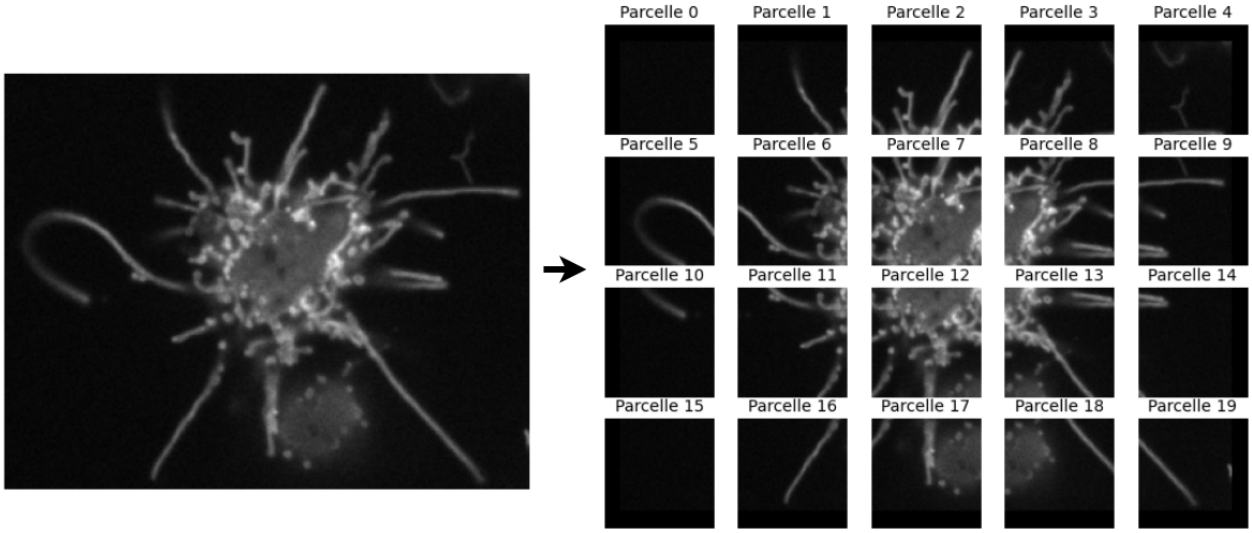


Figure 3.8 – Exemple de division en parcelles avec une tranche du volume Slik3.

Pour combattre le débalancement des classes, on utilise des poids de classes. Sinon, l'arrière-plan risque d'être la seule classe qui a un réel impact lors de l'entraînement et les cytonèmes seraient inconséquents. Les poids sont calculés avec l'équation suivante :

$$W_c = \frac{T}{n} T_c^{-1} \tag{3.4}$$

Où W_c est le poids de la classe c, T est la somme du nombre d'instances pour les n classes et T_c est le nombre d'instances pour la classe c. Dans notre cas, les poids de classes sont : 0.367 pour l'arrière-plan, 38.537 pour les cytonèmes et 4.002 pour les corps de cellules. Dû à une limitation du module Keras (Chollet *et al.*, 2015), les poids sont passés sous forme de volume de poids **L** $\times$ **H** $\times$ **W** qu'on calcule avant l'entraîne-

ment. Les poids de classe sont combinés avec les annotations avec la somme de l'axe **D** avec notre vecteur de poids contenant chaque W_c.

Afin d'expérimenter l'approche SoftSeg, on doit convertir nos annotations binaires en annotations douces. On propose d'exprimer l'ambiguïté d'un voxel à l'aide d'un filtre gaussien avec $\sigma = 1.5$ sur les axes **H** et **W** afin de conserver les contours des cytonèmes. **L** utilise un $\sigma = 1.5(260/120)$ pour prendre compte du changement de distance entre les **voxels**. Un ensemble de données avec des annotations binaires et un ensemble de données avec des annotations floutées sont comparés dans les résultats afin d'identifier le plus performant.

Par ailleurs, l'axe **D** des volumes en niveau de gris est dupliqué 3 fois pour être compatible avec les formats d'entrée de l'architecture U-Net qu'on aborde à la section suivante.

3.3.2 Paramètres d'architecture

Les paramètres utilisés pour l'architecture des modèles 2D et 3D sont dans le tableau 3.2.

Table 3.2 – Paramètres de l'architecture.

Configurations	**U-Net 2D**	**U-Net 3D**
Profondeur du U-Net	4	
Encodeur	**VGG16**	
Activation	ReLU	
Activation de sortie	ReLU	
Séquence pour normaliser	Après l'activation	

Pour les U-Net 2D/3D, on propose d'utiliser une profondeur de 4 et 64 filtres caractéristiques sur la première couche de convolution de sorte à être compatible avec le format d'un encodeur **VGG16**. Les couches d'activation utilisent la fonction d'activation ReLU. Ceci inclut la dernière couche qui classifie chaque **voxel**. Comme proposé dans l'approche SoftSeg, on utilise ReLU au lieu de *softmax* afin de mieux exprimer l'ambiguïté des contours de cytonèmes et de conserver dans la mesure du possible la connectivité entre ses **voxels**.

On utilise un **encodeur pré-entraîné VGG16** sur ImageNet pour avoir un minimum de connaissance lors-

qu'on commence l'entraînement, surtout en raison du fait que nous avons peu de données annotées. Pour combiner le modèle **VGG16** à l'encodeur U-Net, on prend chaque couche avant le **sous-échantillonnage par valeur maximale** et on la connecte à son bloc décodeur qui lui est associé, soit celui au même niveau de profondeur dans l'architecture du U-Net. Le dernier bloc d'encodage du **VGG16** est repris comme bloc milieu et les **couches entièrement connectées** sont retirées. La Figure3.9 illustre les modifications apportées.

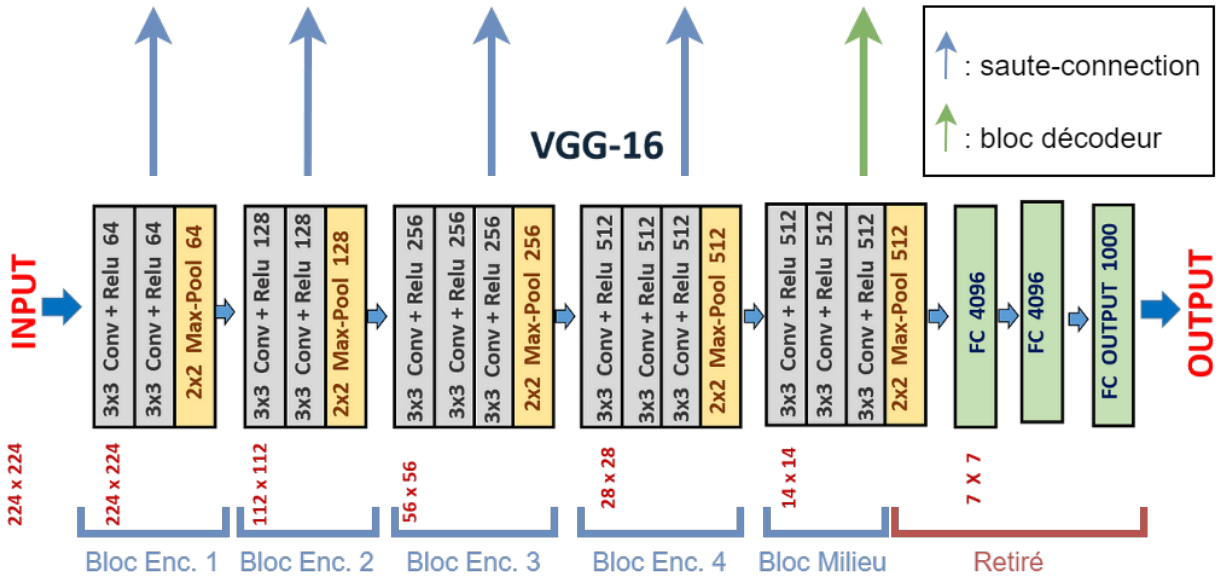


Figure 3.9 – Ajustement de l'encodeur **VGG16** pour U-Net. Les flèches bleues qui vont vers le haut sont les **connexions saute-couches** ajoutées. La figure est adaptée d'un article sur **VGG16** (Khandelwal, 2020).

Puisque l'ensemble ImageNet utilisé pour pré-entraîner notre **encodeur pré-entraîné** est en RGB, l'entrée du décodeur ne pourra pas accepter nos données en niveau de gris par défaut. Une des options pour réduire le nombre de paramètres entraînables dans l'encodeur est de moyenner les poids appris sur la première couche de convolution sur l'axe **D** de sorte à transformer les poids RGB en poids pour des données en niveau de gris. Cette option viendrait diviser par 3 la taille des poids à apprendre. Cependant, on utilise plutôt l'approche de transformer nos volumes en RGB puisqu'on obtient de meilleurs résultats.

Comparativement à l'implémentation d'origine (Ronneberger *et al.*, 2015), les couches de convolution utilisent un **rembourrage** constant à 0 pour le calcul des **cartes de caractéristiques** afin de conserver les dimensions d'entrées pour la segmentation prédite. On modifie aussi la séquence pour faire la normalisation de lot (Çiçek *et al.*, 2016) de sorte qu'elle soit faite suite à l'activation dans les blocs de décodage (Figure3.10).

On obtient une meilleure segmentation avec cet ordre [6].

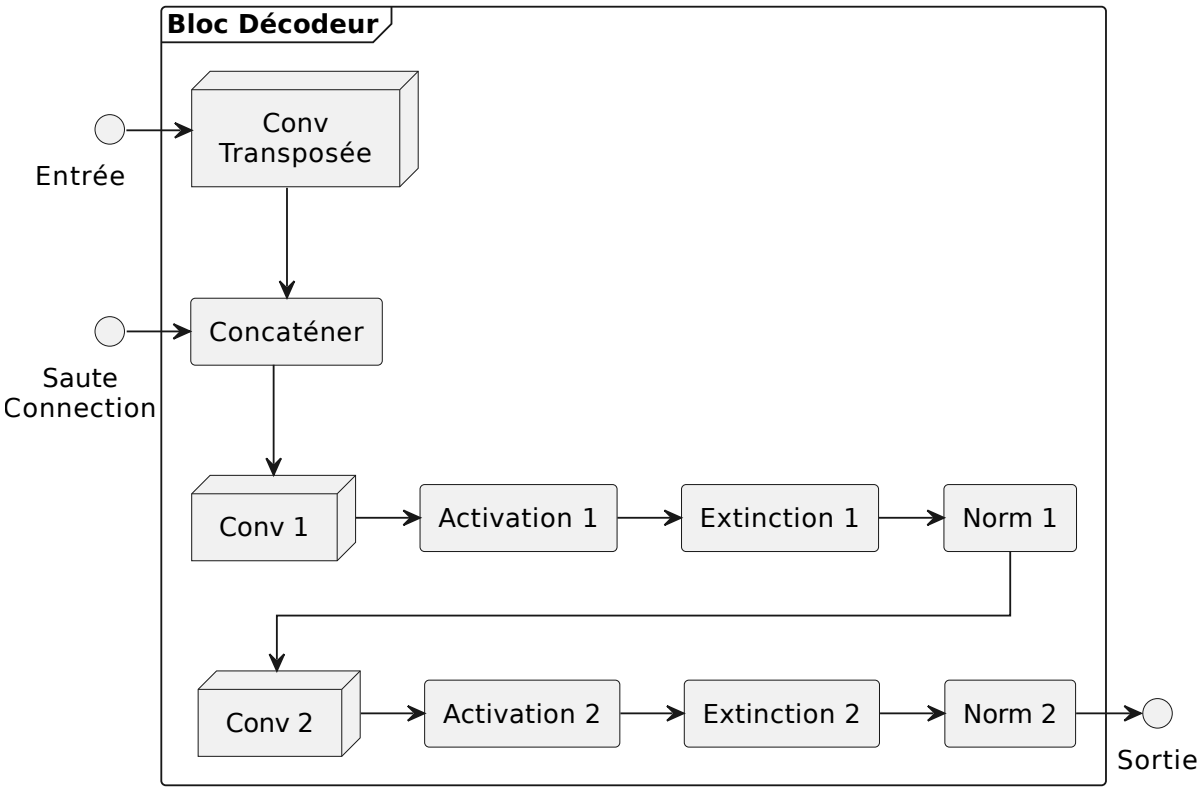


Figure 3.10 – Ensemble des couches pour un bloc de décodage pour un U-Net. *Conv* est pour convolution, *Extinction* est pour **extinction de neurone** et *Norm* est pour normalisation de lot.

3.3.3 Hyperparamètres pour l'entraînement

Le tableau 3.1 contient les **hyperparamètres** utilisés pour l'entraînement des modèles.

Pour pouvoir entraîner avec peu de volumes, il est impératif d'utiliser de l'augmentation de données. On applique des rotations de ± 30 degrés avec un **rembourrage** de type miroir seulement sur les axes **HW**. Il en est de même pour les augmentations d'inversement (*flip*) des axes puisqu'on retrouve des changements d'illumination à travers **L** et que la modification de ceux-ci complexifit l'apprentissage. L'augmentation de **doublons** permet de réutiliser un volume pour sélectionner une parcelle dans une époque. Il est important qu'un volume soit suffisamment réutilisé pour avoir augmenté les chances d'obtenir d'autres classes que l'arrière-plan dans une sous-image en 2D. Il ne s'agit pas vraiment d'un problème en 3D, mais cette

6. Il est important de ne pas suivre une couche de normalisation par une couche d'**extinction de neurone**, sinon on introduit une fuite des poids appris à travers la couche d'**extinction de neurone**

Table 3.3 – Hyperparamètres pour l'entraînement.

Configurations		**U-Net 2D**	**U-Net 3D**
Augmentation de données	**Rotation**	**HW** ± 30 degrés	
	Doublons	16	1
	Inversion	**HW**	
Taille de lot		8	4
Extinction de neurones		40%	
Taux d'apprentissage	**Initialisation**	0.01	0.001
	Mise à jour	**recuite cosinus**	
Fonction de perte		AdaWingClDice	
Arrêt précoce		Patience $= 10$	
Nombre d'époques		100	

transformation permet d'agrandir une époque pour compenser les petites tailles de lots imposées par nos restrictions de mémoire. Pour notre U-Net2D, la taille de lots est de 32 et le taux d'apprentissage de 0.01. En 3D, la taille de lot diminue à 8 pour ne pas dépasser la limite de 12 GB de mémoire GPU, le taux d'apprentissage est 0.001. Le taux d'**extinction de neurone** à travers le décodeur est de 0.4.

Pour la fonction de perte, on propose une nouvelle fonction régressive AdaWingClDice. DiceClDice combine Dice et clDice et cette métrique est très intéressante pour encourager la connectivité des cytonèmes. Ensuite, l'article de Softseg (Gros *et al.*, 2020) obtenait de meilleurs résultats que Dice avec *Adaptive Wing* (AdaWing) grâce à sa meilleure séparation des régions segmentées. AdaWingClDice combine AdaWing et clDice afin d'obtenir des cytonèmes de bonne qualité, soit tubulaires et entiers.

$$L_{AdaWingClDice} = L_{AdaWing} + L_{clDice} \tag{3.5}$$

Notre fonction de perte $L_{AdaWingClDice}$ a comme paramètres $\epsilon = 1$; $\alpha = 2.1$; $\theta = 0.5$; $\omega = 0.001$ pour $L_{AdaWing}$ et 10 itérations pour L_{clDice}. Quoique la fonction L_{clDice} soit conçue pour des formes tubulaires comme les cytonèmes, puisqu'on squelettise seulement sur 10 itérations, on encourage indirectement à conserver le centre des corps de cellules.

L'implémentation de clDice fut en grande partie récupérée du répertoire GitHub de l'autrice (Paetzold, 2021), mais on lui apporte de petites modifications. Le rembourrage pour les filtres utilisés pour la squelettisation semblait être réflexif, causant la formation de lignes sur les contours des images. Ceci est problématique puisqu'on ne s'attend pas à ce que les contours de nos sous-volumes soient précis. Particulièrement en 3D où les sous-volumes sont plus petits et englobent rarement en entier la région d'une classe. On propose d'utiliser un **rembourrage** de valeur constante à 0 pour régler ce problème (Figure3.11). Finalement, on simplifie l'implémentation en remplaçant les filtres minimums par des filtres maximums inversés sur l'image inversée. L'algorithme 3 contient cette modification et il permet d'éviter d'avoir à implémenter un filtre minimum. Bien que notre implémentation soit légèrement modifiée, aucune amélioration significative de segmentation n'a été observée.

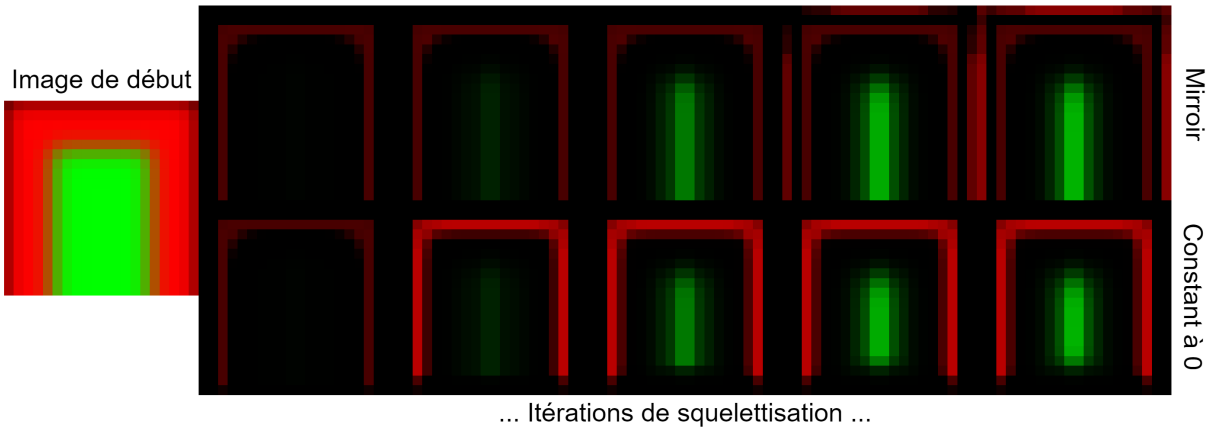


Figure 3.11 – Exemple de squelettisation clDice où l'on compare l'utilisation d'un **rembourrage** miroir et d'un **rembourrage** constant à 0. On observe que le **rembourrage** miroir utilisé dans l'implémentation originale de clDice tend à créer un second squelette sur les rebords.

3.3.4 Transfert d'apprentissage

Entraîner un modèle à partir de zéro prend généralement plus de temps et risque d'être moins performant s'il manque de données. Notre **encodeur pré-entraîné** VGG16 sur ImageNet est un bon début, mais on veut utiliser nos volumes Slik non annotés pour entraîner le décodeur pendant que ce dernier est gelé afin de le mettre à niveau. De cette façon, on peut exposer notre modèle à plus de données que nos 10 volumes annotés lors de l'entraînement. Pour ce faire, on propose de générer des pseudo-annotations en utilisant notre approche par seuillage pour les créer. Celle-ci a déjà été ajustée pour nos données annotées et elle devrait être compatible avec nos données non annotées.

Algorithme 3 Notre squelettisation pour clDice.

procédure Squelettisation(`Image`, `Itérations`)

`Image'` $\leftarrow$ `Filtre Maximum`$(-$`Filtre Maximum`$(-$`Image`$))$ ▷ Ouverture

`Squelette` $\leftarrow$ `ReLU`$($`Image` $-$ `Image'`$)$

tant que $i <$ `Itérations` **faire**

`Image` $\leftarrow -$`Filtre Maximum`$(-$`Image`$)$ ▷ Érosion

`Image'` $\leftarrow$ `Filtre Maximum`$(-$`Filtre Maximum`$(-$`Image`$))$

`Delta` $\leftarrow$ `ReLU`$($`Image` $-$ `Image'`$)$

`Squelette` $\leftarrow$ `Squelette` $+ (1 -$ `Squelette`$) \times$ `Delta`

$i \leftarrow i + 1$

retourne `Squelette`

fin tant que

fin procédure

Le décodeur du U-Net 2D est mis à jour avec nos pseudo-annotations et on complète l'entraînement pour l'ensemble des couches avec nos vraies annotations pour s'assurer de la qualité de l'apprentissage. Ensuite, pour notre U-Net 3D, on propose de convertir les poids appris dans les couches de convolution 2D et les transférer à notre modèle 3D. On fait cela en moyennant les poids 2D sur le nouvel axe **L**. Il manque seulement d'ajuster les nouveaux poids 3D avec nos volumes annotés et l'entraînement est complété. Le transfert des poids 2D à 3D devrait au minimum accélérer l'entraînement du modèle, donc il est recommandé de les utiliser.

3.4 Post-traitement des segmentations

3.4.1 Reconstruction des parcelles

Si les segmentations proviennent d'un modèle U-Net, cela implique que le volume d'origine a été divisé en plusieurs parcelles et qu'il faudra le reconstituer afin de retrouver le format initial du volume. En conservant les informations sur la taille des parcelles, les intervalles, le **rembourrage** utilisé et leur ordre, on peut réassembler les parcelles ensemble. Cependant, il faut gérer les **recoupements** de parcelles. L'algorithme 4 contient la méthode proposée. Les dimensions d'un axe pour le volume reconstruit sont calculées avec l'équation suivante :

$$R_{ax} = P_{ax} + (Nb_p - 1) \times I_{ax} \tag{3.6}$$

Où R_{ax} est la longueur d'un axe ax pour la reconstruction, P_{ax} est la longueur de ax pour les parcelles, Nb_p est le nombre de parcelles sur R_{ax} et I_{ax} est la longueur des intervalles entre les parcelles sur ax (Figure3.12).

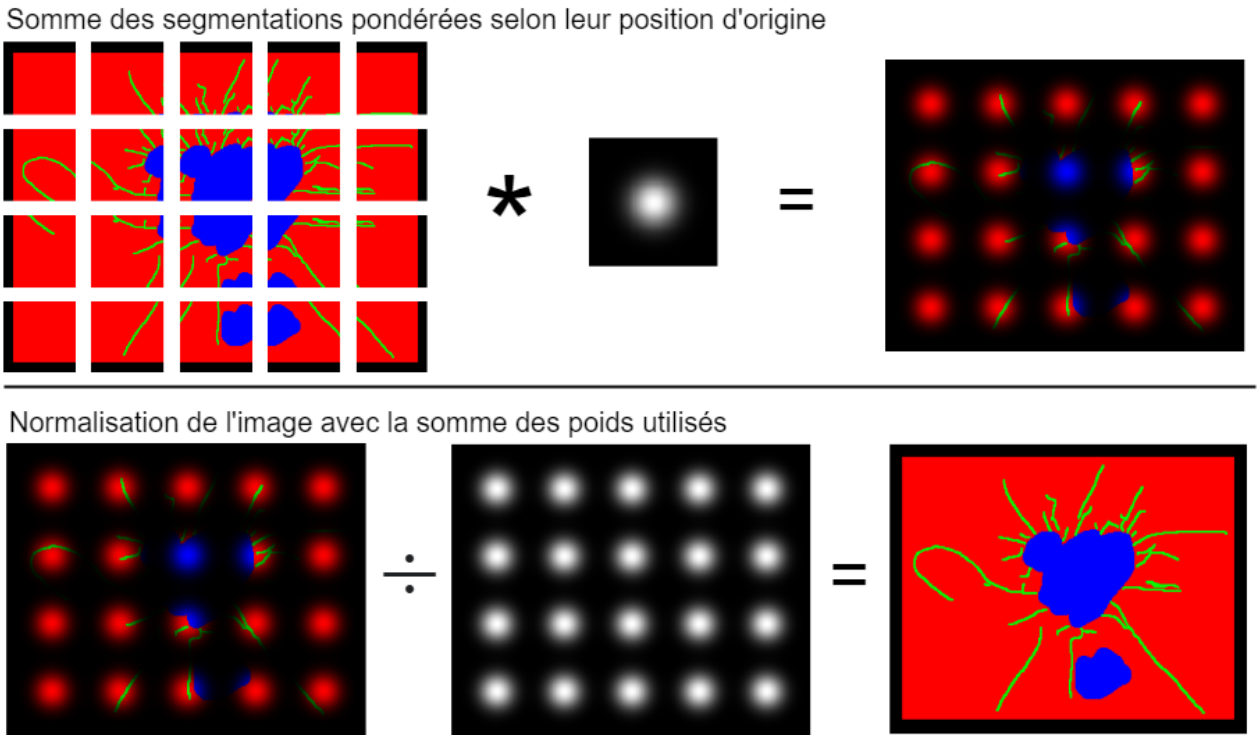


Figure 3.12 – Exemple de réassemblage de parcelles avec une tranche du volume Slik3. Le poids utilisé dans cet exemple pour résoudre les **recoupements** est une distribution gaussienne.

On utilise un filtre 2D gaussien avec $\sigma = 10$ comme matrice de poids pour la fusion d'image, ce qui permet de graduellement combiner les zones d'entrelacements en priorisant les **voxels** plus proches du centre des parcelles grâce à notre gaussienne.

3.4.2 Exploration des cytonèmes

Une fois les classes segmentées, on peut commencer à analyser nos cellules. Pour calculer le nombre moyen de cytonèmes par cellules, nous devons pouvoir identifier à quelle cellule un cytonème appartient. On s'assure d'abord de séparer les corps de cellules en utilisant *watershed*. En calculant la distance de chaque

Algorithme 4 Reconstruction du volume à partir de ses sous-volumes

procédure Reconstruction(`Parcelles`, `Intervalle`, `Ordres`, `Matrice de poids`)

`Reconstruction` ▷ Rassemble les parcelles

`Poids totaux` ▷ Conserve les matrices de poids

$x \leftarrow 0$

`Début x` $\leftarrow x \times$ `Intervalle sur L`

tant que $x <$ `Longueur(Ordre sur axe L)` **faire**

`Début x` $\leftarrow x \times$ `Intervalle sur L`

`Fin x` $\leftarrow$ `Debut x` $+$ `Dimension de parcelles sur L`

$y \leftarrow 0$

tant que $y <$ `Longueur(Ordre sur axe H)` **faire**

`Début y` $\leftarrow y \times$ `Intervalle sur H`

`Fin y` $\leftarrow$ `Debut y` $+$ `Dimension de parcelles sur H`

$z \leftarrow 0$

tant que $z <$ `Longueur(Ordre sur axe W)` **faire**

`Début z` $\leftarrow z \times$ `Intervalle sur W`

`Fin z` $\leftarrow$ `Debut y` $+$ `Dimension de parcelles sur W`

`Parcelle pèsé` $\leftarrow$ `Parcelle à (x,y,z)` $\times$ `Matrice de poids`

`Reconstruction`$[$`Début à Fin de (x,y,z)`$]$ $\leftarrow$ `Parcelle pèsé` ▷ Somme (+=)

`Poids totaux`$[$`Début à Fin de (x,y,z)`$]$ $\leftarrow$ `Matrice de poids` ▷ Somme (+=)

$z \leftarrow z + 1$

fin tant que

$y \leftarrow y + 1$

fin tant que

$x \leftarrow x + 1$

fin tant que

retourne `Reconstruction`/`Poids totaux`

fin procédure

voxel de l'arrière-plan, on obtient notre carte d'élévation et les points d'origine y sont les maximums locaux. Dû à la forme sphérique des cellules, les points d'origine devraient bien approximer le centre des cellules. Ensuite, il s'agit d'associer chaque cytonème avec le corps de cellule le plus rapproché et de calculer la moyenne (Figure3.13).

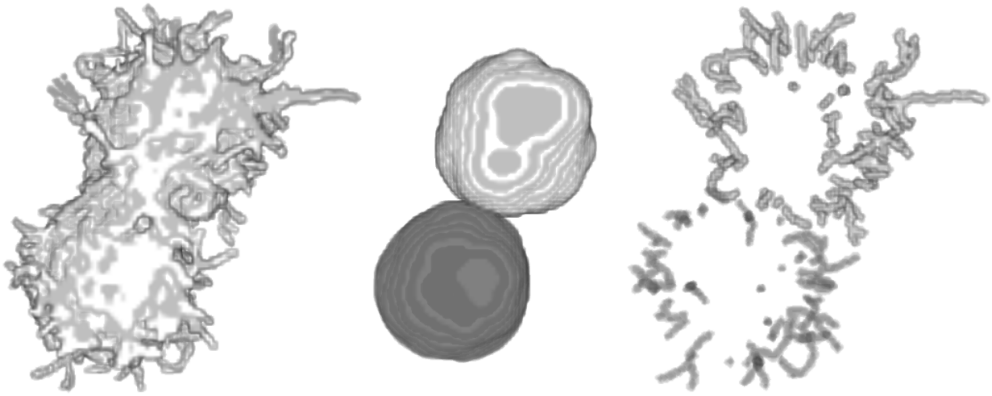


Figure 3.13 – À partir du centre des corps de cellules, *watershed* sépare les instances entrent-elles et les cytonèmes dont le **centroïde** est le plus proche y sont associés. La figure est adaptée de notre article ISBI (Lemieux *et al.*, 2022) avec le volume Slik1.

Pour calculer la longueur d'un cytonème, on propose d'établir un chemin de **voxels** du début du cytonème jusqu'à sa fin. Pour simplifier le parcours d'un cytonème, on propose de squelettiser ceux-ci en 3D. Un squelette morphologique a un seul **voxel** de largeur, donc le chemin à suivre dans un cytonème sera clair. Pour identifier le **voxel** de début pour un cytonème, on prend son point le plus rapproché du corps de cellule qui lui est associé. Pour identifier le **voxel** de fin d'un cytonème, on regarde le voisinage de ces **voxels** avec une connectivité de 8. Si le point a moins de 2 voisins, il est interprété comme la fin d'un cytonème. Chaque squelette de cytonème(s) est transformé en graphe de connectivité de **voxels**. Un **voxel** de squelette devient un nœud et les nœuds dans son voisinage y sont connectés. Pour déterminer le voisinage d'un **voxel**, une connectivité de 8 est utilisée.

Puisque plusieurs cytonèmes peuvent se toucher, il faut pouvoir les distinguer entre eux et on ne peut pas utiliser la même méthode que pour les corps de cellules puisque les cytonèmes sont tubulaires et que les points de départ pour *watershed* ne peuvent pas être bien identifiés. Les cytonèmes peuvent se séparer à un point et s'inter-croiser sur un autre. Pour les naviguer, on propose les règles suivantes. Un ensemble de cytonèmes peut avoir plusieurs points de fin, mais seulement un point de départ. En partant d'un point de fin, on parcourt les nœuds comme DFS, cependant, lorsqu'un nœud est dépilé, il peut être visité à nouveau.

Ceci vise à prendre en compte le cas où il y a deux intersections ou plus du même cytonème sur un chemin.

Pour chaque point sur un chemin, on calcule son orientation avec le produit scalaire des deux **voxels** qui le précède. Pour nettoyer les résultats obtenus, on impose une limite de 1 chemin par point de fin, représentant un cytonème. Pour ce faire, lorsque l'exploration des chemins possibles entre un point de fin et le point de début est complétée, on conserve seulement celui avec l'orientation de **voxels** la plus uniforme. Prenons pour exemple le chemin $\vec{A}$, une ligne entre le point 1 et 2, $\vec{B}$, une ligne entre le point 2 et 3, et $\vec{C}$, une ligne entre le point 2 et 4 (Figure3.14). Si $\vec{A} \cdot \vec{B} < \vec{A} \cdot \vec{C}$, alors le chemin $1 \rightarrow 2 \rightarrow 3$ a des orientations plus uniformes que le chemin $1 \rightarrow 2 \rightarrow 4$ et il sera choisi par notre algorithme.

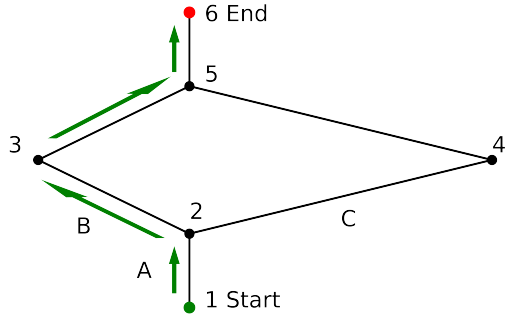


Figure 3.14 – Pour le parcours de graphe, les chemins trouvés seront [1, 2, 3, 5, 6], [1, 2, 3, 5, 4], [1, 2, 4, 5, 6] et [1, 2, 4, 5, 3]. Les chemins qui ne finissent pas sur un point de fin (6) sont rejetés. Le chemin [1, 2, 4, 5, 6] est aussi rejeté puisque seulement 1 chemin par point de fin est permis et [1, 2, 3, 5, 6] (ligne verte) a une orientation plus consistante sur le long de ses coordonnées. La figure est adaptée de notre article ISBI (Lemieux *et al.*, 2022).

3.5 Conclusion

D'abord, ce chapitre présente nos données volumétriques. Nous avons survolé les 3 approches de segmentation qu'on propose pour les traiter. C'est-à-dire l'approche par seuillage qui n'exige aucune annotation et l'approche par U-Net qui demande des annotations. Les paramètres et **hyperparamètres** utilisés par chacun, dont notre nouvelle fonction de perte régressive AdaWingClDice ont été présentés. Nous avons aussi proposé de combiner chacune de nos méthodes pour obtenir le modèle final, soit avec la création de pseudo-annotations et le transfert d'apprentissage 2D à 3D. Finalement, le post-traitement des segmentations pour calculer les métriques de cytonèmes dans la prédiction a été expliqué.

Le chapitre suivant présente l'optimisation de paramètres et d'**hyperparamètres** pour nos trois approches. On y montre les résultats obtenus pour les segmentations et les métriques de cytonèmes. On y discute aussi des avantages, limitations et pistes de recherche pour chacune des méthodes.

CHAPITRE 4
RÉSULTATS EXPÉRIMENTAUX ET DISCUSSIONS

Dans ce chapitre, on discute des résultats de nos expériences pour la segmentation des cytonèmes et des corps de cellules avec nos 3 approches : seuillage global, U-Net 2D et U-Net 3D. Pour évaluer l'entraînement supervisé, nos 10 volumes annotés sont séparés à 60% pour l'entraînement, 20% pour la validation et 20% pour les tests. Une condition est ajoutée pour nos 4 volumes Ctrl de sorte qu'ils soient dans les 3 groupes. Pour l'entraînement auto-supervisé des décodeurs, on sépare les 25 volumes Slik non étiquetés à 80% pour l'entraînement et à 20% pour la validation.

On utilise le coefficient de DiceClDice à l'équation 2.4 afin d'estimer la performance d'un modèle lors de l'entraînement. Le coefficient utilise un ratio de 0.5 entre Dice et clDice. On choisit d'utiliser cette métrique puisqu'elle calcule la validité de la segmentation avec Dice et l'intégrité / connectivité des classes avec clDice. Elle est aussi mieux connue (section2.3.3) comparativement à notre nouvelle fonction de perte $L_{AdaWingClDice}$ proposée.

L'implémentation des modèles U-Net a été faite dans TensorFlow. L'entraînement des modèles a été fait en alternant entre un GPU RTX 2080 11 GB et un GPU RTX 3060 12 GB. En moyenne, l'entraînement d'un modèle U-Net dure 20 minutes en 2D ou une heure en 3D.

On commence en discutant des ajustements de paramètres d'entraînement dans la section 4.1. Dans la section 4.2, on compare entre elles les segmentations des corps de cellules et de cytonèmes entre nos différentes approches. Finalement, on compare nos métriques calculées à celles de B. Rambaud pour les cytonèmes.

4.1 Optimisation de paramètres et d'hyperparamètres

L'outil Weights & Biases (W&B) (Biewald, 2020) a été d'une grande aide pour parcourir automatiquement les paramètres et les **hyperparamètres** considérés pour nos expériences. Dans un monde idéal, chaque expérience aurait utilisé la méthode bayésienne afin d'obtenir une meilleure vue d'ensemble sur l'impact des paramètres. Cependant, pour économiser du temps, on utilise parfois la méthode de grille afin d'accélérer les tests de combinaisons de paramètres à considérer. Les méthodes d'ajustement offertes par W&B sont

montrées dans la figure 4.1.

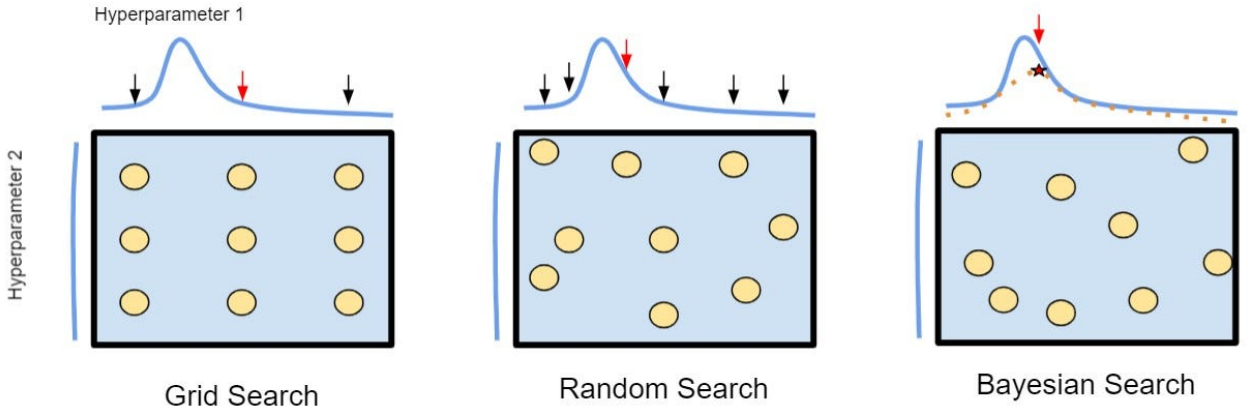


Figure 4.1 – 3 méthodes d'optimisation d'**hyperparamètres**. *Grid Search* est traduit « recherche de grille », où les paramètres sont prédéfinis. *Random Search* est traduit « recherche aléatoire », où les paramètres sont aléatoires, et *Bayesian Search* est traduite « recherche Bayésienne », où l'on converge sur les paramètres aléatoires. La figure fut prise de (Warnes, 2023)

4.1.1 Optimisation de la méthode par seuillage

Dans la section 3.2, on présentait les paramètres configurables pour la méthode par seuillage. Voici les paramètres par défaut utilisés avant notre optimisation (Table4.1).

Table 4.1 – Paramètres par défaut pour l'approche par seuillage.

Configurations		**Méthode par seuillage**
Débruitage	Richardson-Lucy	25 itérations
	NLM	Largeur $= 7vx$; distance $= 11\ vx$
	Filtre médian	Largeur **L** $= 1$; **HW** $= 3$
Filtre d'affinage		$\sigma = 5$; facteur $= 1$
Classification de la cellule		$ro = 1\mu m$; $rc = 0.5\mu m$

On prend la PSF du microscope utilisé pour l'acquisition des volumes avec la méthode de Richardson-Lucy sur 25 itérations. NLM compare des sous-sections de 7 vx^3 à 11 vx de distances. La dimension du filtre médian est $1 \times 3 \times 3$. Le filtre d'affinage a un rayon de 0.6 μm et est appliqué avec un facteur de 1. Les cytonèmes étaient identifiés avec une ouverture qui utilise un **élément structurant** sphérique de 1 μm de rayon (ro). Les corps de cellules étaient remplis avec une fermeture qui utilise un **élément structurant** de

0.5 μm et il n'y a pas de fenêtrage Butterworth.

4.1.1.1 Richardson-Lucy et NLM

Pour le prétraitement de données, l'étape de Richardson-Lucy avec la PSF du microscope confocal joue un rôle majeur dans le débruitage des données (Figure4.2). Cependant, son nombre d'itérations semble peu important. On observe qu'avec une distance et une taille des régions élevées pour NLM, on peut se rapprocher du résultat de Richardson-Lucy. Cependant, en fonction des paramètres utilisés, NLM impacte grandement les performances. La comparaison de sous-régions en 3D est coûteuse en temps, particulièrement sur une grande distance. Une opération de ce genre peut prendre plus d'une heure et elle a peu d'impact suite à notre déconvolution (Figure4.2). C'est pourquoi on la considère comme optionnelle.

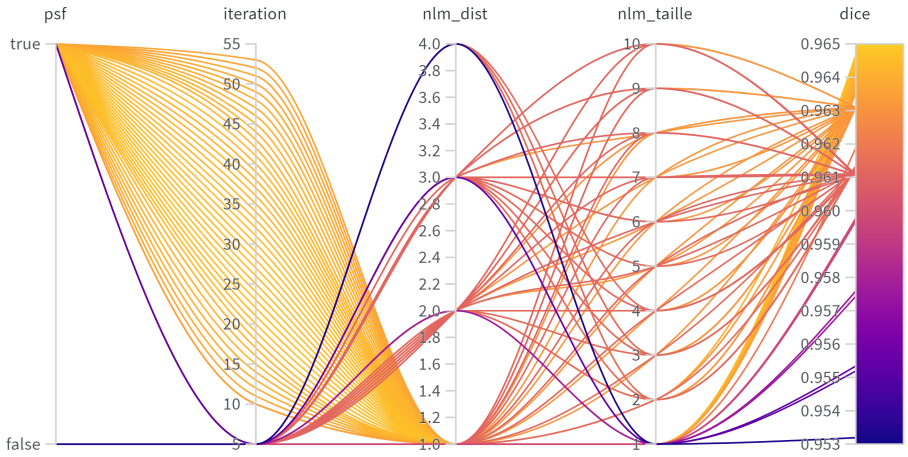


Figure 4.2 – Coefficient de Dice moyen avec notre approche de segmentation par seuillage pour nos volumes annotés. PSF indique si Richardson-Lucy est utilisé. Si oui, Richardson-Lucy est appliqué sur *iteration* itérations. *nlm_taille* réfère à la taille du voisinage qui est comparée sur une distance de *nlm_dist* pour NLM. Exceptionnellement, on utilise une recherche de grille avec *nlm_dist* pour limiter le nombre de valeurs testées.

Dans un scénario où mesurer la PSF n'est pas une option, on propose de l'approximer à l'aide de l'outil *DeconvolutionLab2* (Sage *et al.*, 2017) ou d'utiliser NLM avec une largeur et une distance raisonnablement élevées afin d'augmenter la clarté des régions uniformes.

4.1.1.2 Filtre d'affinage

Notre estimation du σ par défaut pour le filtre d'affinage était très éloignée du résultat final. On prédit qu'un faible σ permettrait de faire ressortir les régions étroites comme les cytonèmes. Cependant, un fort σ aide beaucoup à segmenter les corps de cellules (Figure4.3). Nous verrons dans la section de segmentation que nous avons particulièrement eu de la difficulté à classifier le noyau cellulaire, donc de faire ressortir le corps de cellule plutôt que celui des cytonèmes afin d'arriver à un meilleur coefficient Dice. D'autre part, la différence entre le volume et le filtre gaussien utilise un facteur de 8 avant d'être ajoutée au volume puisque c'est le facteur vers lequel notre recherche bayésienne a convergé (Figure4.3).

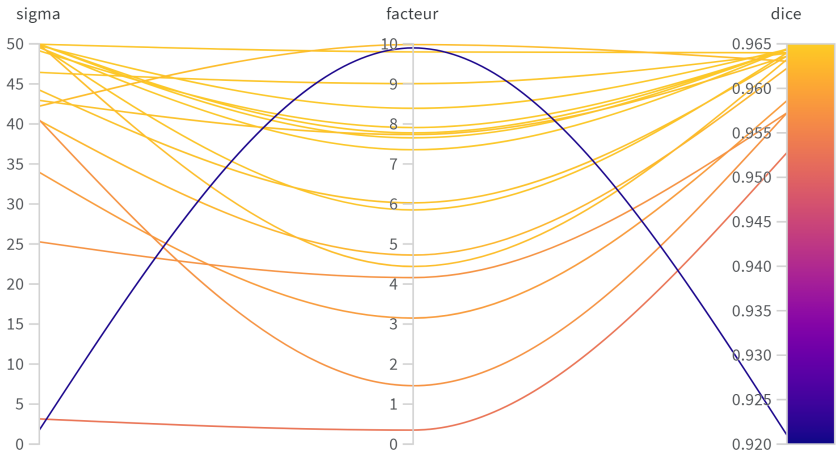


Figure 4.3 – Le coefficient de Dice moyen avec notre approche de segmentation par seuillage pour nos volumes annotés. *sigma* est le σ du filtre gaussien utilisé pour calculer la différence entre l'image et sa version floutée. La différence de chaque **voxel** est ajoutée avec un *facteur* sur l'image d'origine. On fait ici une recherche bayésienne sur chacun des paramètres.

Une amélioration envisageable serait d'utiliser un filtre d'affinage à 2 échelles différentes. Une avec grand σ pour les corps de cellules et une autre avec un petit σ pour les cytonèmes. De cette façon, il serait aussi possible de faire ressortir davantage les cytonèmes comme attendu dans notre hypothèse.

4.1.1.3 Fenêtrage Butterworth

On observe une amélioration des performances de plus de 5% suite à notre fenêtrage Butterworth (Figure4.4). Ces régions étaient inutilement élevées en niveau d'intensité et nuisaient au calcul du seuillage

global. Notre fenêtrage permet de les ignorer en grande partie, mais force celles-ci à être interprétées comme arrière-plan.

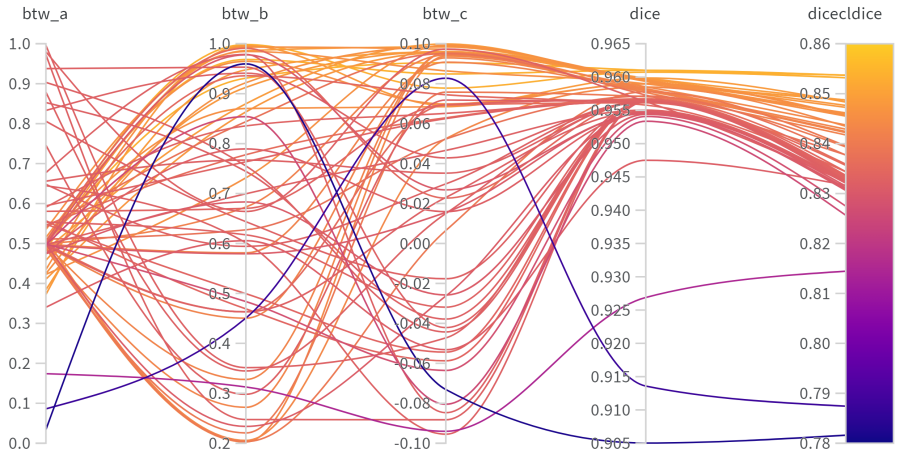


Figure 4.4 – Coefficients de Dice et DiceClDice moyen avec notre approche de segmentation par seuillage pour nos volumes annotés. *btw_a*, *btw_b* et *btw_c* sont équivalents aux paramètres a, b et c de la distribution Butterworth (Éq. 3.2).

Le désavantage de cette méthode réside dans le fait que l'on risque de retirer des régions de volumes qui pourraient contenir des cytonèmes. Il faudrait trouver une meilleure façon de traiter la dégradation linéaire sur l'axe **L**, soit avec une méthode de débruitage qui serait adaptée en fonction de la profondeur pour déconvoluer davantage les régions floues.

4.1.1.4 Filtre médian et classification

Pour le filtre médian, on utilise une longueur de plus d'un **voxel**, puisque l'axe **L** contient beaucoup moins d'information que **HW**. On risquerait de perdre des détails importants, mais la Figure4.5 (md_ax0) démontre plutôt l'inverse. Ceci est probablement dû au bruit de Poisson élevé sur cet axe, et un filtre médian sur **L** de taille 3 à 5 qui viendra compenser pour celui-ci. Même si on perd beaucoup d'information avec la médiane, on y trouverait plus de bruit que d'information pertinente pour un seuillage.

On observe qu'un ro à 1 μm est la taille qui se généralise le mieux pour notre ensemble de données. Un ro à 1 permet de classifier toutes les régions plus grandes qu'une sphère avec ce rayon similaire à celui d'un cytonème. À moins d'augmenter à un ro de 1.5, les débuts des cytonèmes sont souvent interprétés

comme un corps de cellule, mais on inclut plus de bruit. On peut en filtrer la majorité en fonction de leur taille, mais un ro de 1 est notre juste milieu. D'ailleurs, notre méthode par seuillage est plutôt grossière pour identifier les petites régions comme les cytonèmes puisqu'elle ne tient pas en compte les formes comme nos méthodes par apprentissage machine. En prenant un grand **élément structurant** pour la restauration des corps de cellules (rc de 1.5 μm), on observe un meilleur coefficient de Dice. Il est mieux d'avoir un petit ro à 1 μm. La valeur optimale est la plus petite possible, soit la distance des **voxels** sur **L**.

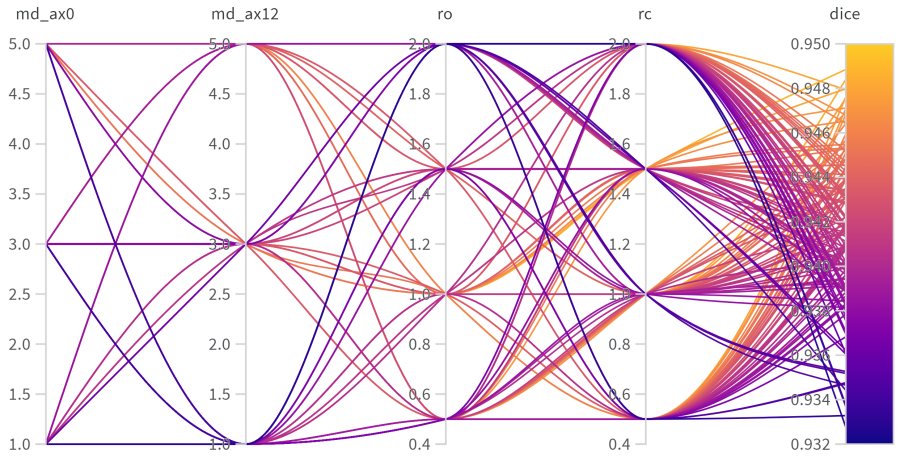


Figure 4.5 – Coefficient de Dice moyen avec notre approche de segmentation par seuillage pour nos volumes annotés. *md_ax0* réfère à la taille du **noyau** sur l'axe **L**, tandis que *md_ax12* est pour **H** et **W**. *ro* est pour le rayon de l'**élément structurant** de l'ouverture morphologique qui sépare les cytonèmes des corps de cellules. *rc* est pour le rayon de l'**élément structurant** utilisé pour la fermeture morphologique pour remplir les corps de cellules. Les paramètres sont explorés avec une recherche en grille pour réduire les combinaisons envisageables.

Dans le cas où l'on chercherait à optimiser notre méthode par seuillage pour de nouvelles données, rc et ro ont un grand impact sur la qualité de la classification et devraient être les premiers paramètres à optimiser afin de bien segmenter les cytonèmes sans trop diminuer la qualité des corps de cellules.

4.1.1.5 Paramètres optimisés de l'approche par seuillage

Suite à nos optimisations de paramètres dans la section 4.1.1, notre version finale de l'approche par seuillage utilise les paramètres dans la table 4.2.

Table 4.2 – Paramètres optimaux pour l'approche par seuillage.

Configurations		**Méthode par seuillage**
Débruitage	Richardson-Lucy	50 itérations
	NLM	Largeur $=$ N/A ; distance $=$ N/A
	Filtre médian	Largeur **L** $= 1$; **HW** $= 3$
Filtre d'affinage		$\sigma = 50$; facteur $= 8$
Classification de la cellule		$ro = 1\mu m$; $rc = 1.5\mu m$
Fenêtrage Butterworth		$a = 0.5$; $b = 0.95$; $c = 0.09$

4.1.2 Optimisation des paramètres U-Net

Dans la section 3.3, on présentait les paramètres et les **hyperparamètres** configurables pour U-Net. La Table4.3 contient les paramètres et les **hyperparamètres** sur lesquels nos expériences ont commencé. Les optimisations 2D restent valides pour la version 3D de notre modèle [1], donc on réutilise les mêmes paramètres 2D en 3D.

4.1.2.1 Prétraitement des données

Pour le type d'annotation utilisée, les annotations douces [2] étaient préférables aux annotations binaires sans même utiliser une fonction de perte régressive. D'autre part, on estimait que normaliser les intensités de **voxels** entre 0 et 1 serait suffisant, mais on observe une légère augmentation des performances avec la standardisation qui calcule leur écart-type entre -1 et 1 (Figure4.6). On utilise aussi Richardson-Lucy sur 25 itérations comme pour l'approche par seuillage.

Les tailles des parcelles de volume à l'entrée des modèles ont été ajustées empiriquement au cours de nos expériences dues à des contraintes d'espaces mémoire à notre disposition. Lors de l'optimisation, deux hypothèses ont été utilisées. La première, l'axe **L** des parcelles devrait être aussi grand que **L** pour prendre compte de l'ensemble des dégradations linéaires observées sur l'axe. La deuxième, un axe de parcelle n'a pas intérêt à dépasser la taille des volumes annotés, sinon cela implique d'utiliser du **rembourrage** lors du calcul de la perte.

1. À l'exception des tailles de parcelles, la taille de lots et l'augmentation de doublons sont dues à des contraintes de mémoire
2. floutées avec un filtre gaussien $\sigma = 1.5$

Table 4.3 – Paramètres et **hyperparamètres** par défaut pour U-Net.

Configurations		**U-Net 2D**	**U-Net 3D**
Prétraitement	**Échelle**	Normalisée	
	Sous-volume	$1 \times 96 \times 96 \times 1$	$48 \times 96 \times 96 \times 1$
	Annotation	Binaire	
	Poids	$C_y = 38.54; C_o = 4; \overline{C_o \cup C_y} = 0.37$	
Augmentation de données	**Rotation**	**HW** ± 30 degrés	
	Doublons	48	1
	Inversion	**HW**	
Taille de lot		32	3
Profondeur du U-Net		4	
Encodeur		U-Net	
Séquence pour normaliser		Avant l'activation	
Activation		ReLU	
Activation de sortie		Softmax	
Extinction de neurones		10%	
Taux d'apprentissage	**Initialisation**	0.001	
	Mise à jour	**recuite cosinus**	
Fonction de perte		Dice	
Arrêt précoce		Patience $= 10$	
Nombre d'époques		100	

Généralement, on observe de meilleurs résultats lorsque l'axe **L** est petit et les axes **HW** sont maximisés. Ceci infirme notre première hypothèse et confirme la deuxième puisque l'utilisation du **rembourrage** sur l'axe **L** à 48 diminuait les performances comparativement à une longueur de 38 [3]. L'axe **D** à 1 permet de réduire la taille des données de 3 comparativement à rouge, vert et bleu (RGB), mais le coefficient Dice des entrées en couleurs est ± 0.04 meilleur que pour les entrées grises.

Pour équilibrer les classes, on utilise des poids de classes implémentées directement dans les coefficients et les fonctions de perte lors de l'entraînement et de la validation. C_y est le poids d'une instance cytonèmes,

3. Pour Slik4 qui a l'axe **L** le plus court (FigureA.1)

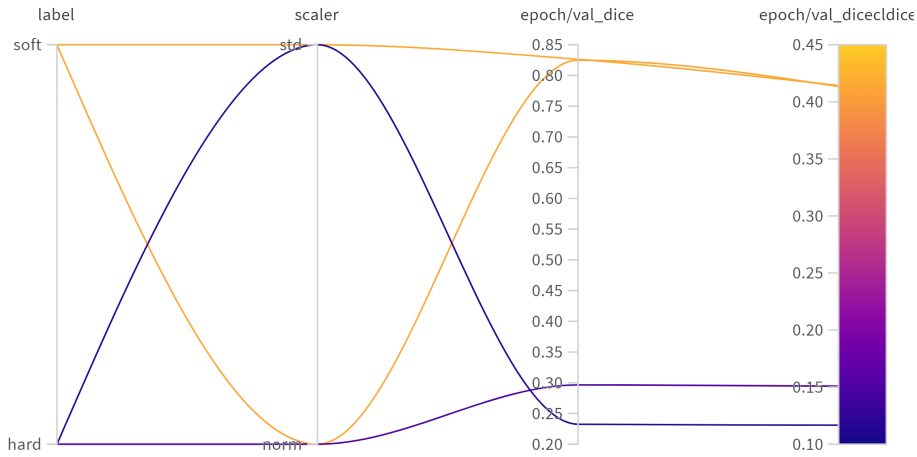


Figure 4.6 – Coefficient de Dice et DiceClDice moyen avec notre approche de segmentation par U-Net 2D pour 60% des volumes annotés. *label soft* et *label hard* réfèrent au type d'annotation douce ou binaire. *scaler* est pour la méthode d'ajustement des valeurs de **voxels** à l'entrée au modèle entre normalisation et standardisation.

C_o est le poids d'une instance de corps et $\overline{C_o \cup C_y}$ est pour les instances d'arrière-plan. Basé sur une expérience préliminaire, on segmente presque exclusivement de l'arrière-plan sans l'utilisation de poids de classe (Figure4.7). Les classes sont si déséquilibrées qu'on obtient un score Dice qui semble bon en prédisant seulement de l'arrière-plan.

Malheureusement, le module TensorFlow ne supporte pas l'utilisation de poids de classe pour des données volumétriques, donc l'alternative proposée est d'utiliser des **poids d'instances** qui permettent d'attribuer un poids individuel à chaque **voxel**. Cependant, les points d'instances n'ont pu être transmis aux fonctions de pertes par leurs paramètres lors de l'entraînement. C'est pourquoi la solution finale inclut les poids de classe dans l'implémentation de nos fonctions de perte.

4.1.2.2 Augmentation de données

À l'exception des doublons, les paramètres d'augmentation de données n'ont pas été expérimentés. Sans dupliquer nos données d'entraînement, la qualité de l'apprentissage lors d'une époque est trop aléatoire. Nos volumes contiennent beaucoup d'arrière-plan et lors d'une époque, il se peut que seulement des sous-volumes contenant uniquement des vxs de cette classe. En moyenne, sélectionner plusieurs sous-régions

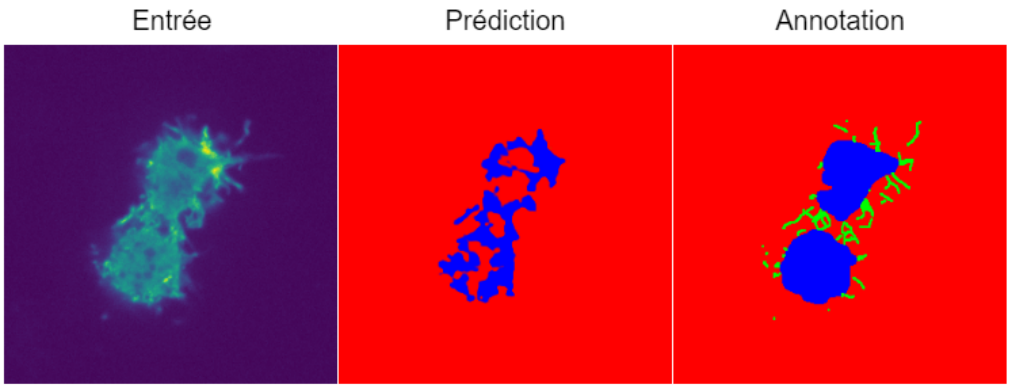


Figure 4.7 – Piètre prédiction du volume de test Slik1. Le modèle 2D utilisé pour cette prédiction avait été entraîné sans poids de classe, par conséquent, il a convergé presque exclusivement vers la classe d'arrière-plan. Il en est tel que les cytonèmes ne semblent même plus être considérés lors de la segmentation.

par volume donne une meilleure répartition des classes par époque et réduit la probabilité d'un tel scénario. Il ne s'agit plus d'un problème en 3D puisqu'on couvre une grande partie de l'axe **L**, permettant de compenser pour la petite taille de lots que nos restrictions de mémoire imposent en augmentant le nombre de **pas de gradient** à chaque époque (Keskar *et al.*, 2016). Sinon, sans doublons, une époque se finit en 3 pas, ce qui entraîne trop de variances dans les données d'entraînement entre les époques. On conclut qu'avec un ensemble de données si petit, les doublons permettent d'amoindrir les valeurs de perte aberrantes lors de l'entraînement.

4.1.2.3 Architecture

Une contrainte imposée afin de pouvoir bien comparer un encodeur traditionnel avec un encodeur **VGG16** est qu'on impose à nos U-Net une profondeur de 4 et 64 filtres caractéristiques pour la première couche de convolution pour que les **connexions saute-couches** combinent des dimensions similaires. D'ailleurs, quoiqu'un encodeur **VGG16** pré-entraîné augmente le nombre de paramètres entraînables, on observe une amélioration de ± 0.05 avec Dice contre l'encodeur originel (FigureA.4) (FigureA.3).

Pour l'ordre entre l'activation et la normalisation de lot, on utilise d'abord la méthodologie standard (Ioffe et Szegedy, 2015), mais l'ordre *Activation* $\Rightarrow$ *Extinction* $\Rightarrow$ *Normalisation* performait légèrement mieux avec ± 0.01 Dice de plus que l'ordre *Extinction* $\Rightarrow$ *Activation* $\Rightarrow$ *Normalisation*. Les premières itérations des modèles U-Net utilisaient une couche d'activation ReLU après les couches de convolutions et une couche d'ac-

tivation Softmax pour la classification des **voxels**. En itérant sur chaque combinaison envisagée, on constate que le type d'activation suite aux couches de convolutions n'est pas vraiment conséquent et qu'elle dépend surtout de la fonction de perte et d'activation finale (Figure4.8). La meilleure combinaison obtenue est ReLU avec celle de ReLU normalisée pour la prédiction.

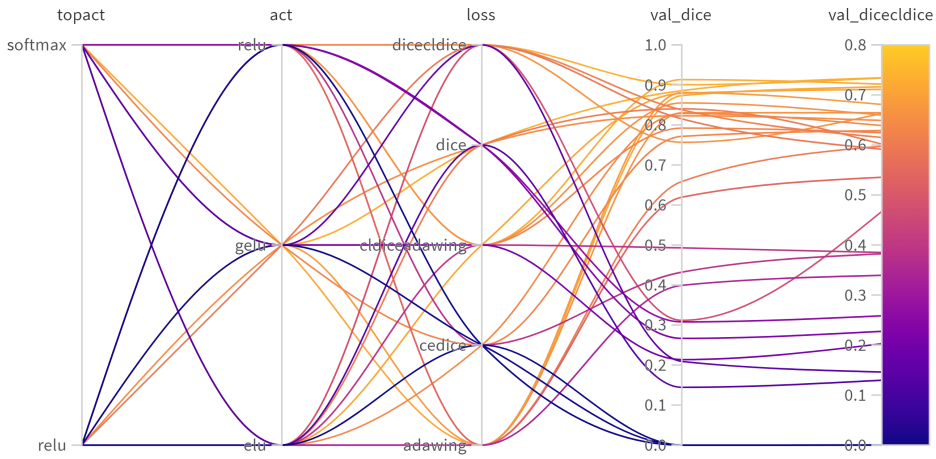


Figure 4.8 – Coefficient de Dice et DiceClDice moyen avec notre approche de segmentation par U-Net 2D pour 60% des volumes annotés. *topact* est pour l'activation finale avant la prédiction. *act* est l'activation utilisée suite aux couches de convolutions et *loss* est la fonction de perte.

On spécifie la couche d'**extinction de neurone** puisqu'il est important de ne pas introduire une fuite des poids de neurones en faisant une normalisation suivie d'une **extinction de neurone**. On parle bien d'une fuite puisque tous les poids d'une couche seront pris en compte lors de la normalisation de lot des poids, donc ils auront un impact même s'ils sont exclus par la couche d'**extinction de neurone** par la suite. Par ailleurs, pour ne pas trop influencer de façon aléatoire les paramètres testés, le taux d'**extinction de neurone** était à 0.1 au départ de l'optimisation, mais on observe qu'un taux inférieur à 0.15 et supérieur à 0.45 ont tendance à diminuer les performances du U-Net (Figure4.9).

4.1.2.4 Apprentissage

Le taux d'apprentissage peut être mis à jour continuellement en fonction de l'époque courante afin de faciliter la convergence du modèle. Suite à des expériences préliminaires pour cette recherche, il a été déterminé qu'il serait préférable d'utiliser une fonction de mise à jour du taux d'apprentissage. Celles-ci ont été faite avec une fonction linéaire, une fonction **recuite cosinus**, une fonction *one-cycle policy* et sans aucune fonc-

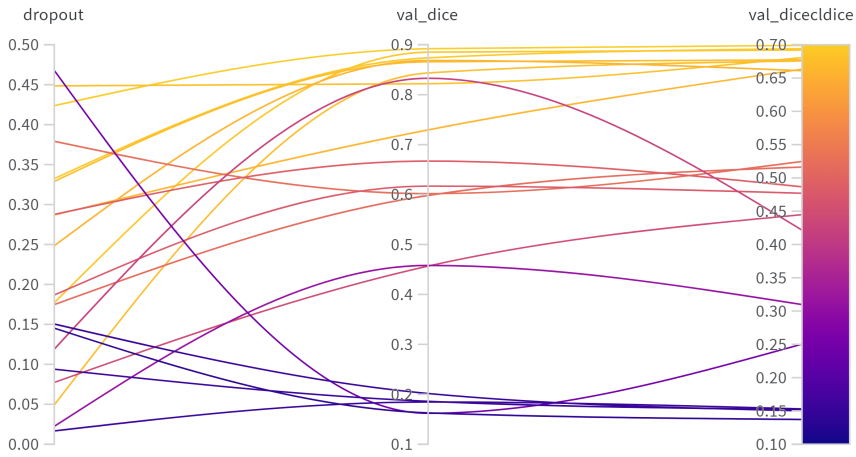


Figure 4.9 – Coefficient de Dice et DiceClDice moyen avec notre approche de segmentation par U-Net 2D pour 60% des volumes annotés. *dropout* est le taux d'**extinction de neurone** suite aux couches de convolutions.

tion. Sans utiliser de fonction, on finit souvent dans un minimum local non optimal. En commençant par un échauffement (*warmup*) du taux d'apprentissage, on vise à réduire le nombre d'époques nécessaires pour finir l'entraînement, ce qui exclut l'utilisation d'une simple fonction linéaire. La fonction **recuite cosinus** a été choisie plutôt que la fonction OCP puisqu'elle est incluse par défaut dans Keras et qu'elle semble obtenir des résultats tout aussi consistants. Le taux d'apprentissage initial pour nos tests était de 0.01. Comme règle du pouce, celui-ci était divisé par 10 pour chaque transfert d'apprentissage effectué. Dans notre cas, « un transfert » réfère à l'utilisation d'un décodeur pré-entraîné et au transfert des poids 2D à 3D.

Une première tentative d'optimisation du taux d'apprentissage initiale a été faite (Smith, 2017) basée sur plusieurs entraînements d'une époque. On visait à explorer l'entraînement par la super-convergence qui vise à utiliser le plus haut taux d'apprentissage sans y introduire de surapprentissage (Smith et Topin, 2018). Les valeurs initiales de taux d'apprentissage ont été explorées sans succès. Suite à l'ensemble de nos optimisations pour U-Net, nous constatons qu'utiliser une activation finale ReLU et une fonction de perte *Adaptive Wing* (AdaWing) donne des résultats trop variables pour établir des conclusions si tôt dans l'entraînement. Les paramètres finaux pour le taux d'apprentissage sont basés sur nos observations générales, particulièrement en 2D. Il faut attendre jusqu'à la fin de l'entraînement pour bien voir les performances de la fonction de taux d'apprentissage.

La vitesse et la qualité de l'entraînement de modèle dépendent primordialement du taux d'apprentissage. Nos optimisations ont été faites en 2D et ont été rapidement adaptées pour en 3D. Il faudrait encore davantage de recherche sur ceux-ci pour notre entraînement en 3D puisque la majorité de son optimisation est basée sur nos observations en 2D. En poussant davantage nos recherches pour ce paramètre de sorte à trouver les taux optimaux pour nos données, notre entraînement devrait bénéficier grandement de ceux-ci.

Le nombre d'époques était largement surestimé et il a été réduit à moins de 40 pour éviter de faire du surapprentissage sur les volumes d'entraînement. Afin d'éviter de faire l'ensemble des époques lorsque notre modèle n'est plus productif, on y ajoute un arrêt précoce. Si le coefficient de DiceClDice ne s'améliore par pour l'ensemble de validation après 10 époques, on arrête l'entraînement, ce qui était particulièrement utile initialement lorsqu'on entraînait sur 100 époques.

Finalement, nous utilisions au départ une perte Dice pour son adaptabilité à différents problèmes en segmentation. Suite à nos tests (Figure4.8), on observe que ce ne sont pas toutes les combinaisons de fonction de perte et d'activation qui performent bien ensemble. La meilleure combinaison utilise ReLU avec notre fonction de perte régressive clDiceAdaWing, suivit de près par la combinaison GELU et DiceClDice.

4.1.2.5 Paramètres optimisés de l'approche par U-Net

Suite à nos optimisations de paramètres et d'**hyperparamètres** dans la section 4.1.1, notre version finale de l'approche par seuillage utilise les paramètres dans la table 4.4.

4.2 Comparaison des méthodes de segmentation

La table 4.5 compare les résultats des différentes approches qu'on propose entre elles. Pour l'évaluation des performances, il aurait été préférable d'utiliser *Leave One Out cross-validation* (LLOCV), mais on se contente d'utiliser les volumes Ctrl1 et Slik1 que l'on a dédiés pour nos tests afin de rentrer dans les délais de cet ouvrage. L'objectif principal étant d'obtenir des cytonèmes suffisamment bien définis pour être parcourus en longueur afin de calculer leur longueur dans la prochaine section. Le deuxième objectif étant d'avoir des corps de cellules raisonnablement fiables pour pouvoir y associer les cytonèmes trouvés.

On divise nos résultats en 2 colonnes. La première colonne est pour les coefficients obtenus avec les parcelles des volumes réservés pour les tests. La deuxième colonne est pour pouvoir bien comparer les résul-

Table 4.4 – Paramètres et **hyperparamètres** optimaux pour U-Net.

Configurations		**U-Net 2D**	**U-Net 3D**
Prétraitement	**Échelle**	Standardisée	
	Sous-volume	$1 \times 192 \times 192 \times 3$	$16 \times 112 \times 112 \times 3$
	Annotation	Douce	
	Poids	$C_y = 38.54; C_o = 4; \overline{C_o \cup C_y} = 0.37$	
Augmentation de données	**Rotation**	**HW** ± 30 degrés	
	Doublons	48	16
	Inversion	**HW**	
Taille de lot		16	3
Profondeur du U-Net		4	
Encodeur		**VGG16**	
Séquence pour normaliser		Après l'activation	
Activation		ReLU	
Activation de sortie		ReLU normalisée	
Extinction de neurones		30%	
Taux d'apprentissage	**Initialisation**	0.0001	
	Mise à jour	**recuite cosinus**	
Fonction de perte		$L_{AdaWingClDice}$	
Arrêt précoce		Patience $= 10$	
Nombre d'époques		40	

tats 2D au 3D en reconstruisant un volume où on conserve seulement une classe par **voxel**, soit celle avec la plus haute probabilité. Chacune des reconstructions utilise un **noyau** gaussien avec $\sigma = 10$ pour gérer les **recoupements**.

Notre U-Net 3D, où on entraîne d'abord l'encodeur avec nos données non annotées, obtient les meilleurs coefficients Dice et DiceClDice. En deuxième place se trouve le U-Net 2D, où l'on fait le même entraînement de l'encodeur, mais en 2D pour ensuite transférer les poids en 3D. En troisième place vient ensuite la méthode par seuillage et, en dernier, Ilastik qui démontrait précédemment des signes de surapprentissage lors de l'annotation des volumes section3.1.2.

Table 4.5 – Comparaison entre nos modèles de segmentation proposés

	Parcelles		Volumes reconstruits	
	Dice	DiceClDice	Dice	DiceClDice
U-Net 3D	0.9038	0.6698	0.9284	0.7711
U-Net 2D	0.9241	0.6871	0.9230	0.7526
Seuillage	—	—	0.9237	0.7406
Ilastik	—	—	0.8240	0.6054

Avec notre meilleure approche de segmentation, U-Net 3D, on observe que même si l'acquisition est floue, on peut encore raisonnablement bien classifier les **voxels** dans un cas comme Slik1, mais on perd davantage de cytonèmes dans ces régions (Figure4.10). Cependant, pour un cas extrême comme Ctrl1, les régions floues d'arrière-plan ont tendance à être classifiées comme corps cellulaire sur une grande superficie.

Une fois les volumes segmentés, avant même de collecter les métriques pour les cytonèmes, on observe que Ctrl1 exprime beaucoup moins de cytonèmes que Slik1. Les 2 volumes contiennent 2 instances de cellules relativement proches qu'il faudra séparer. On souligne que le troisième globe (plus petit) classifié comme corps de cellule dans Ctrl1 n'est pas considéré comme une troisième cellule, car il ne contient pas de noyau cellulaire.

4.2.1 Avantages et limitations

L'avantage unique à l'approche par seuillage est qu'elle ne requiert pas d'entraînement. En faisant une étude qualitative des résultats, on peut éviter d'avoir à annoter un ensemble de données. Elle peut rapidement être configurée pour un volume, et ce, sans avoir à le diviser en parcelle. Cependant, cette approche ne peut pas être généralisée à l'ensemble de nos volumes. Elle n'est pas assez flexible pour considérer les différentes tailles de cytonèmes et de corps de cellules lors d'opérations morphologiques (Figure4.5). Plusieurs paramètres doivent être ajustés pour obtenir une segmentation acceptable.

De plus, cette approche n'a rien pour distinguer le lamellipode des cytonèmes. Ce dernier peut être acceptable lorsqu'il est intégré avec le corps d'une cellule puisque la forme reste plus ou moins sphérique. Cependant, dans le cas des cytonèmes, ceux-ci peuvent perdre leur forme tubulaire et devenir inutilisables pour la squelettisation. On constate aussi que beaucoup des cytonèmes segmentés sont des faux négatifs

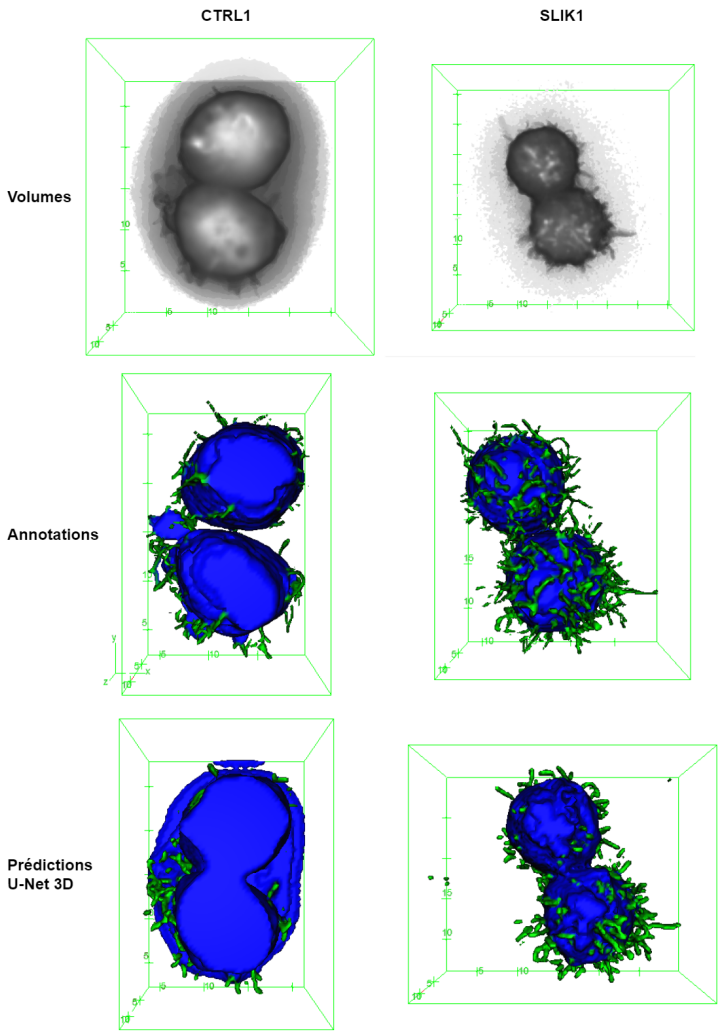


Figure 4.10 – Représentation en maillage triangulaire de la segmentation des volumes de tests Ctrl1 et Slik1 avec l'approche U-Net 3D. Les axes indiquent la distance en μm dans le volume à partir du **voxel** (0,0,0). La taille des axes varie visuellement entre les volumes parce que la méthode de visualisation utilisée retire les sections d'axes avec aucun maillage.

puisque les corps de cellules sont remplis de trous et le rayon de la fermeture binaire est trop grand pour les **voxels** de corps de cellule soit correctement classifié (Figure4.11).

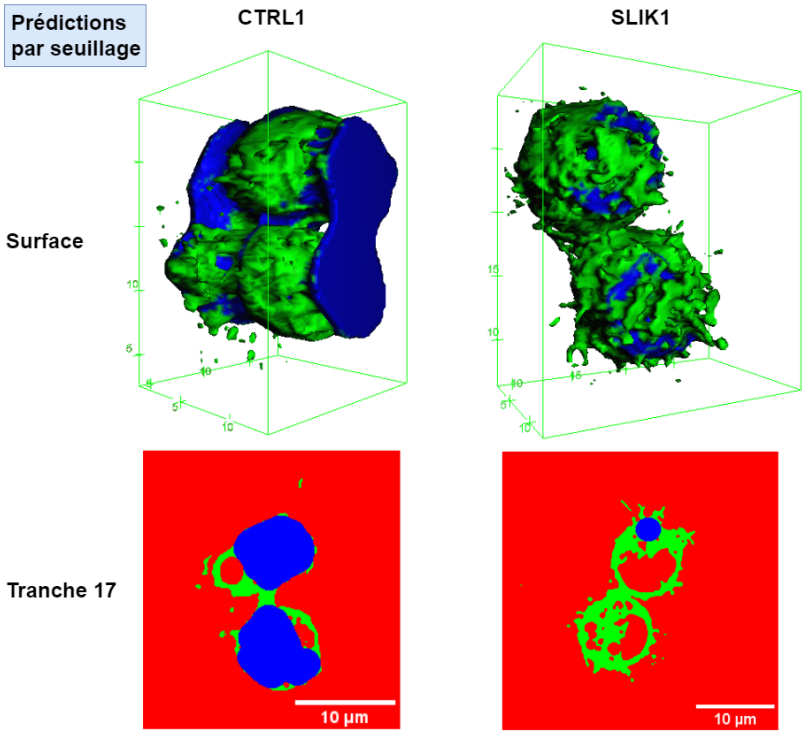


Figure 4.11 – Représentation en maillage triangulaire de la segmentation des volumes de tests Ctrl1 et Slik1 pour l'approche par seuillage. Les axes indiquent la distance en μm dans le volume à partir du **voxel** (0,0,0). Surface réfère au maillage triangulaire et tranche 17 à une image sur l'axe de profondeur **L**. La taille des axes varie visuellement entre les volumes parce que la méthode de visualisation utilisée retire les sections d'axes avec aucun maillage.

Ilastik pour sa part demande d'être entraîné, mais son interface graphique est intuitive et interactive à un point tel qu'il semble être plus facile d'optimiser sa segmentation d'un volume que la méthode par seuillage. Cependant, elle a aussi de la difficulté à se généraliser pour l'ensemble de nos volumes. Cette approche peut distinguer des formes à différentes échelles et conserve donc mieux la forme tubulaire des cytonèmes. Elle arrive aussi à mieux combler les noyaux cellulaires, un problème malheureusement présent dans chacune de nos méthodes. Cependant, le modèle de forêts aléatoires entraîné a beaucoup de difficultés avec les régions de volume floues qu'on semble incapable de classifier comme arrière-plan (Figure4.12).

La méthode par U-Net obtient par une avance considérable de meilleurs résultats et elle se généralise mieux aux volumes de tests. Cependant, les modèles sont plus complexes à entraîner, et une carte GPU est pré-

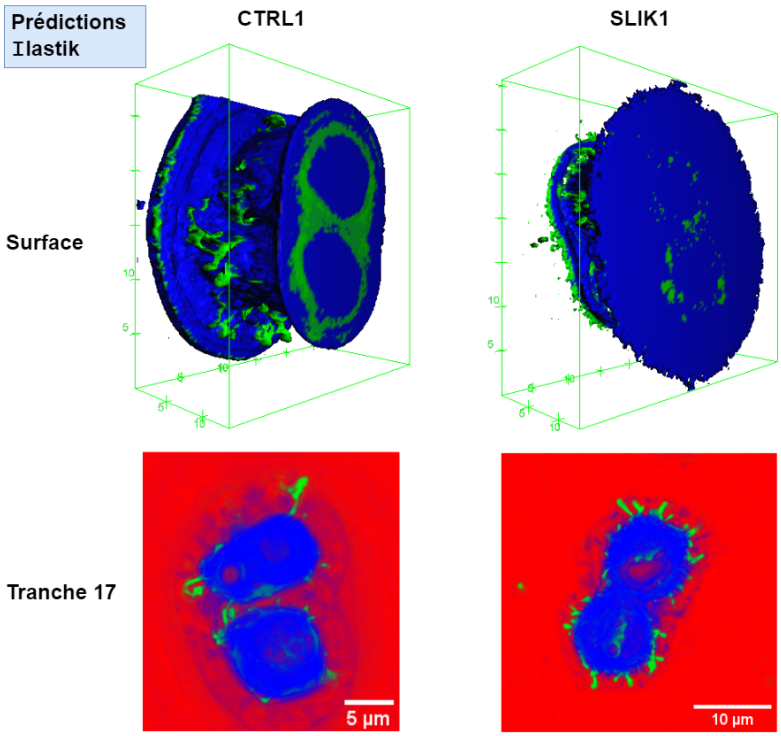


Figure 4.12 – Représentation en maillage triangulaire de la segmentation des volumes de tests Ctrl1 et Slik1 avec l'aide d'Ilastik. Les axes indiquent la distance en μm dans le volume à partir du **voxel** (0,0,0). Surface réfère au maillage triangulaire et tranche 17 à une image sur l'axe de profondeur **L**. La taille des axes varie visuellement entre les volumes parce que la méthode de visualisation utilisée retire les sections d'axes avec aucun maillage.

férable pour les entraîner dans un délai raisonnable. D'ailleurs, à partir de TensorFlow 2.11, l'entraînement GPU n'est plus supporté multiplateforme [4]. L'entraînement supervisé demande aussi de mettre en place un ensemble de données suffisamment grand afin de bien généraliser ses segmentations. Cette méthode demande aussi davantage de mémoire, imposant la division des volumes en parcelles et la reconstruction de ceux-ci pour segmenter un volume entier.

Comparativement à un U-Net 3D, l'entraînement 2D est beaucoup plus simple et rapide. Cependant, puisqu'on n'y considère pas l'axe de profondeur **L**, certaines tranches de volume sur cet axe peuvent être complètement différentes. Notre U-Net 3D segmente sensiblement de la même façon, mais il nous assure une meilleure connectivité des classes sur **L**. On observe que U-Net arrive à mieux classifier les régions floues

4. Windows n'est plus supporté à moins de passé à travers un sous-système Linux (WSL).

comme arrière-plan et à mieux segmenter nos classes cytonèmes que les autres méthodes explorées (Figure4.10) (Figure4.13). On note que Ilastik arrive à mieux remplir les corps de cellules.

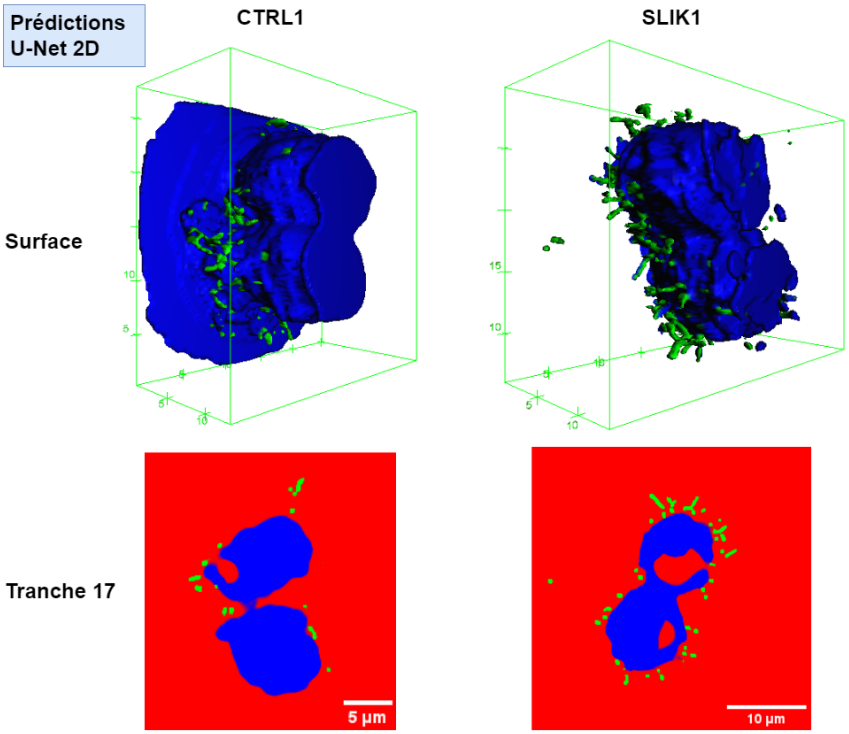


Figure 4.13 – Représentation en maillage triangulaire de la segmentation des volumes de tests Ctrl1 et Slik1 avec notre U-Net 2D. Les axes indiquent la distance en μm dans le volume à partir du **voxel** (0,0,0). Surface réfère au maillage triangulaire et tranche 17 à une image sur l'axe de profondeur **L**. La taille des axes varie visuellement entre les volumes parce que la méthode de visualisation utilisée retire les sections d'axes avec aucun maillage.

En passant d'un U-Net 2D à 3D, on arrive à réduire le temps d'entraînement 3D de 10 époques et à obtenir une segmentation tout aussi bonne. On pourrait aussi possiblement apporter des connaissances uniquement possibles à obtenir en 2D. D'où l'intérêt de l'entraînement non supervisé du décodeur de notre U-Net 2D qui permet d'utiliser des volumes non annotés. Les volumes annotés sont plus nombreux et volumineux que nos volumes annotés et exigent beaucoup trop de temps pour les utiliser avec un U-Net 3D. Cependant, on constate que nos pseudo-annotations avec la méthode par seuillage ne sont pas suffisamment bonnes pour donner une valeur ajoutée aux performances de segmentation d'un U-Net 2D. Avec ou sans pré-entraînement du décodeur, aucune amélioration n'a été observée.

Par ailleurs, lors de l'entraînement des modèles U-Net, les opérations d'augmentation de données utilisées

sont plus simplistes que désirées puisqu'elles ont dues être partiellement implémentées. Elles sont limitées à des opérations basiques pour ne pas complexifier le projet davantage. PyTorch est utilisé par les modules réputables comme MONAI (Cardoso *et al.*, 2022) et TorchIO (Pérez-García *et al.*, 2021). Cependant, ce projet a commencé en 2D avec TensorFlow sans savoir que le support 3D était beaucoup plus limité. Une première approche a été faite avec Albumentations (Buslaev *et al.*, 2020) et Volumentations (Solovyev *et al.*, 2022), mais des problèmes de comportement [5], de stabilité du module [6] et d'absence de documentation étaient problématiques avec Volumentations (FigureA.6).

4.2.2 Améliorations et pistes de recherche

Un problème récurrent rencontré avec nos différentes approches est que les segmentations des corps de cellules sont souvent incomplètes à cause du noyau cellulaire. Augmenter la taille des **noyaux** de convolution de 3 à 5, ou utiliser un *stride* de 2 pour U-Net n'a pas fonctionné non plus. Une piste de solution serait d'avoir des caractéristiques pour différentes échelles comme le fait Ilastik qui pourraient être combiner à l'aide d'un **ré-échantillonnage avec remise ensembliste** afin de mieux remplir ceux-ci.

S'il est possible d'augmenter la mémoire et la taille de notre ensemble de données, on pourrait augmenter la taille des **noyaux** de convolutions dans nos U-Net **noyaux** à 31 et plus pour possiblement mieux considérer la région autour des **voxels** à classifier (Ding *et al.*, 2022). Sinon, en poursuivant la continuité et la réutilisation de nos modèles, on peut prendre notre U-Net 3D et remplacer sa dernière activation par un **transformeur visuel**. Les couches de notre modèle U-Net serait gelé et le **transformeur visuel** serait responsable de corriger les lacunes observées dans nos résultats, notamment le remplissage de corps cellulaire et les régions floues.

Pour capitaliser sur nos transferts de poids 2D à 3D, il faudrait donner une meilleure valeur ajoutée que notre méthode d'entraînement pseudo-supervisé proposée. On juge qu'en utilisant Ilastik pour créer nos pseudo-annotations, nous pourrions dépasser les performances d'un entraînement puisqu'elle offre une meilleure fiabilité (Figure4.12) que les cytonèmes et les corps de cellules segmentés dans les régions non floues comparés à la méthode par seuillage (Figure4.11). Non seulement les transferts d'apprentissage économiseraient du temps d'entraînement, mais ils permettraient au modèle d'apprendre davantage sur les

5. Transformation *GridDropout* s'applique seulement sur l'une des diagonales d'un volume

6. Perte de la majorité des **voxels** lors de certaines rotations

types de cellule à considérer. On rappelle que ceci est important puisqu'on entraîne seulement nos modèles avec 8 volumes, dont 2 pour validation. Utiliser davantage d'augmentation de données devrait faciliter l'entraînement de nos modèles. Soit en passant de TensorFlow à PyTorch ou en utilisant Albumentations pour notre entraînement 2D, ajoutant ainsi une autre valeur possiblement bénéfique au transfert de poids vers 3D.

Finalement, pour la distribution de notre modèle à nos collaborateurs, il serait préférable d'avoir une interface graphique pour faire la segmentation des modèles. Ilastik propose maintenant l'utilisation de modèle sur le site *Model Zoo* (BioImage.IO, 2023) et pourrait être utilisé si on y envoie notre U-Net 3D. L'équipe du projet Ilastik travaille activement sur une solution afin de permettre l'entraînement de tel modèle [7].

4.3 Comparaison de la quantification des métriques

Pour l'analyse des métriques cytonèmes trouvées, notre collaborateur B. Rambaud a calculé les longueurs de cytonèmes pour les volumes de test Ctrl1 et Slik1. À partir d'une vue orthogonale du bas de l'axe de profondeur **L**, les coordonnées 2D de début et fin de cytonèmes ont été identifiées afin de pouvoir calculer à la main la distance entre celles-ci (Figure1.2). En premier lieu, une étude qualitative sera faite en comparant celles-ci à celle obtenue en 3D avec les annotations. On utilise ensuite les annotations afin d'évaluer notre méthode de calcul de métriques cytonèmes sans dépendre de la qualité de notre segmentation dans la section précédente.

On observe dans Figure4.14 que même à partir d'une segmentation optimale, soit celle de B.Rambaud, plusieurs cytonèmes sont perdus suite à notre exploration. Un ratio de 0.5837 des cytonèmes est conservé pour Ctrl1 et un ratio de 0.6849 pour Slik1. Ceci peut être dû à plusieurs facteurs. Un cytonème trop court, ou à tortuosité improbable, sera filtré après coup. Par exemple, si un cytonème contient moins de 5 **voxels**, ou si un **voxel** sur son chemin est dans une direction perpendiculaire à la dernière direction explorée.

De plus, si un cytonème n'arrive pas à satisfaire le critère d'avoir un point de début et un point de fin, il ne sera jamais exploré. Ce qui arrive souvent si la fin du cytonème est assez proche du corps de cellule pour être considérée comme un autre point de début. Par ailleurs, plusieurs cytonèmes annotés n'ont pas de point de début et de fin puisque ceux-ci avaient été faits sur une seule image du volume qui ne contenait

7. Version 1.4.0.post1 (10 novembre 2023)

pas l'ensemble des cytonèmes.

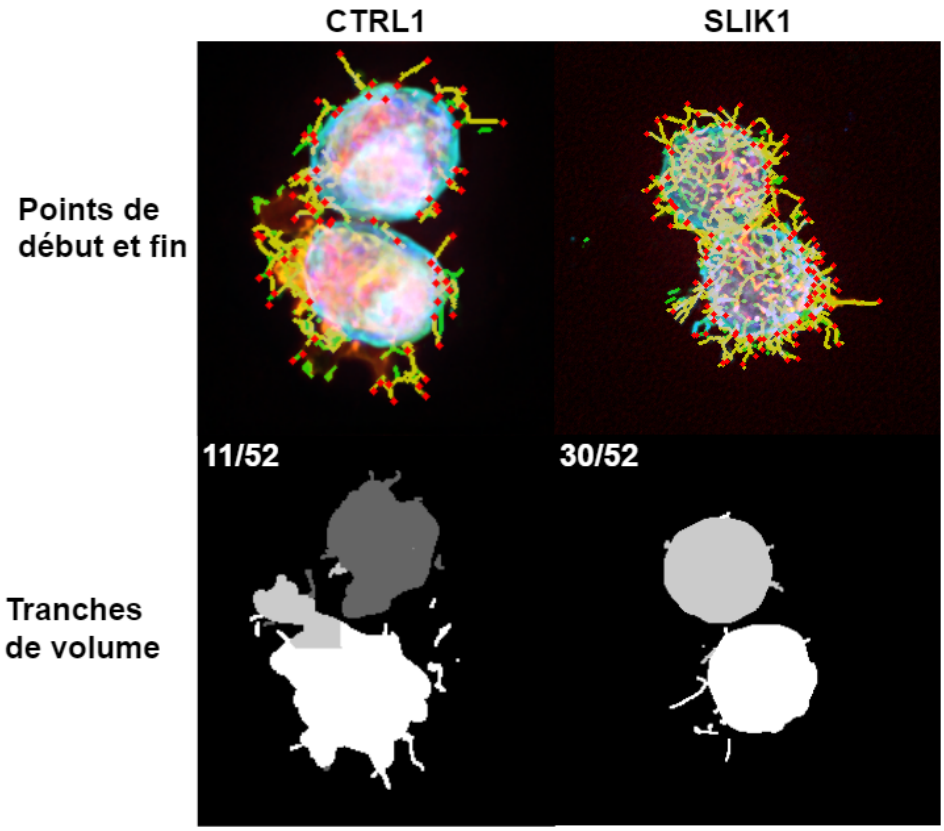


Figure 4.14 – Images 2D des volumes de tests Ctrl1 et Slik1. En jaune sont les cytonèmes conservés suite à l'exploration des cytonèmes segmentés en vert. En rouge sont les points de début et fin de cytonèmes annotés par B.Rambaud. Pour visualiser la profondeur **L**, on varie la couleur du volume en fonction de celle-ci. Sur 2 tranches de volumes, on montre quel cytonème est associé à quel corps de cellule.

4.3.1 Avantages et Limitations

Approcher la problématique de longer un cytonème avec une approche de graphe simplifie l'implémentation d'une telle tâche et ne demande aucun entraînement. Quoiqu'on offre quelques corrections aux problèmes rencontrés avec les données annotées, on constate qu'il faudrait beaucoup plus de conditions d'exception pour bien performer avec d'autre volume. Le problème avec cette méthode est qu'elle a été conçue en tenant pour acquis que les segmentations suivent 2 règles.

La première règle est que les corps de cellules soient de forme sphérique et raisonnablement grands pour facilement les distinguer des cytonèmes. Pour l'association des cytonèmes entre les différents corps de

cellules, on observe que pour Slik1, la séparation de sphère fonctionne comme attendu puisque les corps de cellules sont sphériques. Cependant, pour Ctrl1 où le corps de cellule est concave à certains endroits, la séparation interprète plus de corps de cellules qu'il y en a. Ceci viendra brouiller les métriques de moyenne entre les cellules telle que la longueur moyenne de cytonèmes par cellule. De plus, les cytonèmes entre les corps de cellules sont classifiés plus ou moins aléatoirement puisqu'ils sont aussi proches d'un corps que de l'autre.

La deuxième règle est que le parcours de graphe dépend fortement de la squelettisation et que celle-ci donne de mauvais résultats si les cytonèmes ne sont pas suffisamment tubulaires. Par exemple, avec la méthode par seuillage (Figure4.11), les métriques de cytonèmes peuvent être erronées. La squelettisation créera de faux chemins qui risquent d'être inutilisables en raison de leur tortuosité trop élevée pour être conservée.

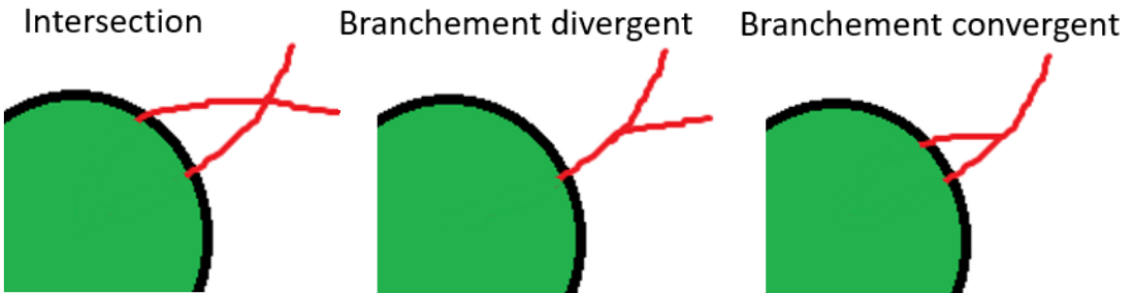


Figure 4.15 – Exemples des croisements de cytonème sur lesquels notre algorithme de parcours de cytonème fut implémenté.

Le défi de prendre en compte les différents types de croisements de cytonèmes a été plus complexe qu'envisagé. Nous avons surtout considéré les cas simples avec une cellule (Figure4.15). Par exemple, une intersection avec une autre cellule n'est pas prise en compte. Dans ce cas, l'un des cytonèmes sera associé à la mauvaise cellule et il y aura trois coordonnées de fin de cytonèmes au lieu de deux. Notre choix de point de début et de fin de cytonème n'est pas assez générique et exclu souvent plus du tiers des cytonèmes segmentés. Notamment, lorsque le lamellipode est classifié comme cytonème, ce qui introduit beaucoup d'ambiguïté pour le choix du point de début du cytonème.

Ces limitations viennent remettre en question la fiabilité des métriques calculées. Cependant, les biais qu'elles introduisent demeurent constants pour chaque volume. Ceci permet d'avoir rapidement une première opinion qu'il faudrait ensuite vérifier si elle est intéressante. Une métrique qui demeure valide serait

le ratio de **voxels** cytonèmes sur cellules. On observe que Slik à généralement un ratio plus élevé que Ctrl avec 0.17 contre 0.1 à partir des volumes annotés.

4.3.2 Améliorations et pistes de recherche

Pour l'instant, les métriques de distances sont calculées à partir d'un script Python et inscrites dans un fichier CSV pour un volume donné. Afin de simplifier l'utilisation de notre solution, on voudrait une interface graphique qui n'impose pas à nos collaborateurs de l'UdeM d'avoir à configurer un environnement de travail. La création d'une image Docker a été considérée, mais notre projet dépend de plusieurs modules Python peu maintenus. Lors de la création de l'image, on risque de rencontrer des problèmes comme un conflit de dépendance et une version des modules qui n'est plus supportée des modules. On considère soit une extension ImageJ ou la compilation d'un exécutable à partir de notre script sur le(s) système(s) d'exploitation qu'ils préfèrent.

Une bonne quantification des métriques dépend fortement de la qualité des segmentations données en entrée. Puisque l'étape de segmentation est séparée de la méthode de quantification, nos collaborateurs ne seront pas restreints aux méthodes de segmentation proposées dans la section précédente si celles-ci ne s'avèrent pas suffisamment générique pour leurs futures acquisitions de cellules. Si nos collaborateurs venaient à utiliser nos méthodes de segmentations, on recommanderait de rendre les noyaux cellulaires fluorescents aussi pour l'acquisition afin de faciliter la tâche.

D'autre part, la gestion des croisements de cytonèmes ne devrait pas être traitée dans notre algorithme de parcours de cytonèmes. Son implémentation est déjà complexe et il est difficile de prendre en compte tous les scénarios de croisement de cytonèmes envisageables. Une fois nos classes segmentées, on aimerait explorer l'utilisation d'un modèle de segmentation d'instances pour possiblement mieux les distinguer et les associer à leur corps de cellule. Cependant, il faudrait chercher comment interpréter un **voxel** où un croisement a lieu avec plusieurs instances de cytonèmes.

4.4 Conclusion

Dans ce chapitre commence en présentant nos optimisations de méthodes. Suite aux ajustements de paramètres et d'**hyperparamètre**, on démontre qu'on peut atteindre un DiceClDice de 0.7711 avec notre U-Net 3D, suivit de U-Net 2D, seuillage et Ilastik. Il est souligné que même si Ilastik performe moins bien, l'outil

reste une bonne option pour sa puissante interface graphique qui permet d'ajuster nous-mêmes l'entraînement en temps réel. Finalement, on évalue de façon qualitative notre méthode de calcul de métrique à partir d'un volume de cellule segmenté. On constate que cette approche est un peu naïve, car elle suppose que les corps de cellules sont sphériques et elle n'arrive parfois pas à les séparer adéquatement. Cependant, elle a le potentiel d'être utilisée comme premier coup d'œil pour comparer les volumes Ctrl et Slik puisqu'elle applique les mêmes biais dans les 2 cas.

CONCLUSION

Cette recherche explore 3 approches de segmentation de cytonèmes, avec et sans intelligence artificielle. En utilisant une méthode de traitement d'images de bas niveau, on constate que le noyau cellulaire dans les corps de cellules cause beaucoup de problèmes pour faire des opérations morphologiques. Puisque cette méthode utilise une ouverture morphologique pour faire la distinction entre les cytonèmes et les corps de cellules, elle est peu fiable. On se tourne donc davantage vers U-Net en 2D et 3D pour segmenter les corps et cytonèmes de cellule. Cette approche se généralise beaucoup mieux avec nos données de tests, cependant elle demande davantage de préparation et nous a imposé de mettre en place un ensemble de données annotées pour l'entraînement des modèles.

Afin d'assurer une continuité dans le projet, nous proposons un entraînement itératif afin d'obtenir notre U-Net 3D ; l'approche avec les meilleures performances. L'approche par seuillage permettrait une première passe pour la segmentation et elle pourra être utilisé pour créer des pseudo-annotations afin de pré-entraîner un U-Net 2D dont les poids seraient ensuite transférés en 3D. Cependant, les pseudo-annotations n'ont pas amélioré les performances du U-Net 2D et donc la seule valeur ajoutée du transfert d'apprentissage entre 2D et 3D est de diminuer le nombre d'époques nécessaires pour l'entraînement 3D.

L'objectif de pouvoir segmenter les corps de cellules et leurs cytonèmes à partir d'un volume confocal fut atteint pour les cellules où le noyau cellulaire ne touche pas à l'arrière-plan. Les méthodes de segmentation proposées devront pouvoir gérer ces cas afin d'être considérées fiables ou améliorer notre méthode de remplissage de cellules utilisée suite à nos segmentations.

De plus, nous proposons aussi une nouvelle méthode d'exploration de cytonèmes afin de pouvoir passer **voxel** par **voxel** et d'ainsi être en mesure de calculer des métriques comme la longueur d'un cytonème. Pour ce faire, on squelettise les cytonèmes segmentés pour les transformer en graphe et les traverser de façon similaire à l'algorithme DFS. Cette méthode est encore au stade préliminaire de développement, mais elle permet tout de même d'estimer l'expression des cytonèmes.

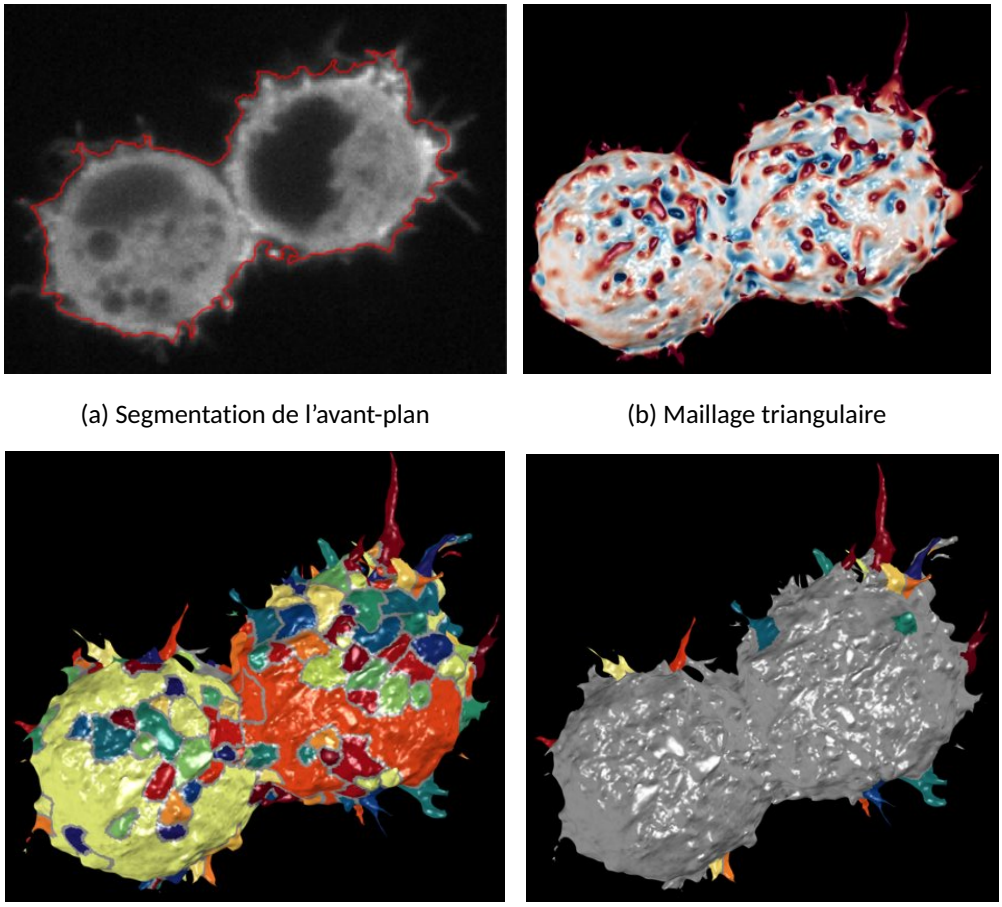
Puisque nos données demeurent privées au moment de cette rédaction, l'objectif d'extraire des métriques sur l'expression de cytonèmes peut seulement être évalué par nos collaborateurs. Cependant, nous jugeons que les métriques collectées par notre implémentation dépendent fortement de la qualité de la segmen-

tation et de la séparation des cellules. Cette limitation demande donc la supervision d'un expert avant d'utiliser notre méthode.

En conclusion, cette recherche permet à partir d'un volume confocal de segmenter les corps de cellules et leurs cytonèmes pour ensuite obtenir un premier avis sur le niveau d'expression de cytonèmes parmi les cellules. En arrivant à dépasser les performances d'Ilastik, nous sommes en mesure de démontrer que les segmentations par U-Net 3D sont efficaces. En améliorant la fiabilité des métriques collectées à partir des segmentations, nous sommes d'avis que notre approche serait prometteuse pour la recherche de développement de cytonèmes.

ANNEXE A

ANNEXE



(a) Segmentation de l'avant-plan (b) Maillage triangulaire

(c) Séparation de la surface (d) Région classifiée comme cytoneme

Figure A.1 – Classification des régions sur la surface de la cellule Slik1 avec l'outil u-shaped3D. Plusieurs cytonèmes sont perdus à l'étape lors de la segmentation initiale.

Table A.1 – Dimensions des volumes annotés.

	Ctrl1	Ctrl2	Ctrl3	Ctrl4	Slik1	Slik2	Slik3	Slik4	Slik5	Slik6
Axe **L**	52	52	39	39	52	52	39	38	39	39
Axe **H**	234	293	200	341	298	282	235	309	422	344
Axe **W**	213	423	240	330	294	294	297	367	370	360

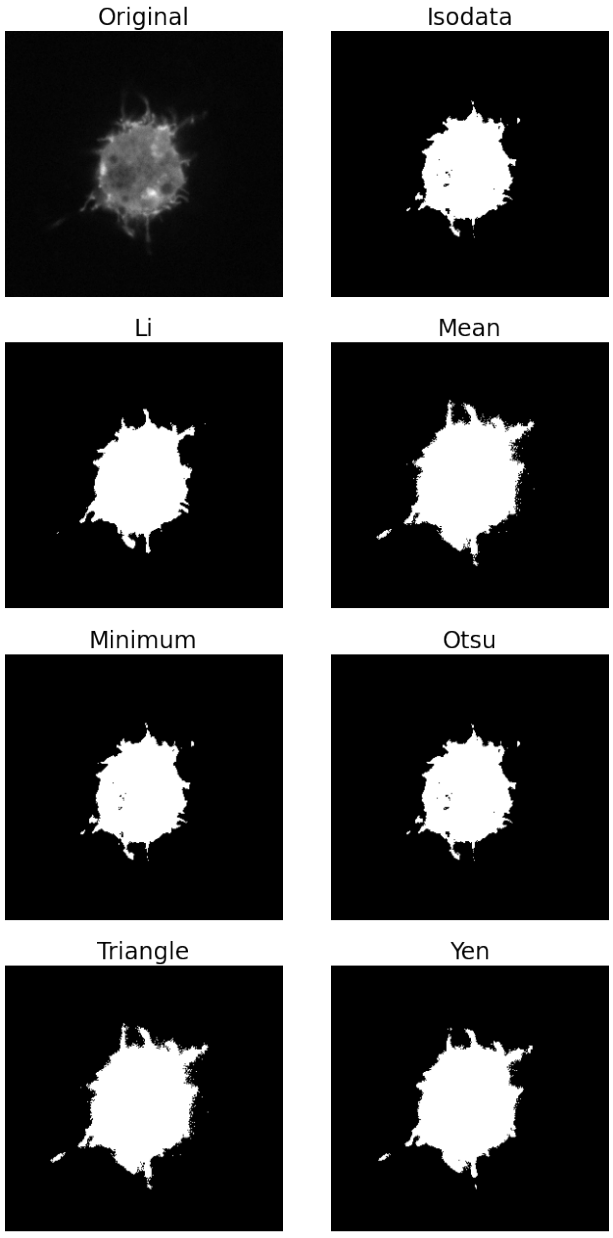


Figure A.2 – Segmentation par méthode de seuillage globale. La méthode de Li donne généralement une meilleure précision pour les cytonèmes et n'exagère pas trop les corps de cellules.

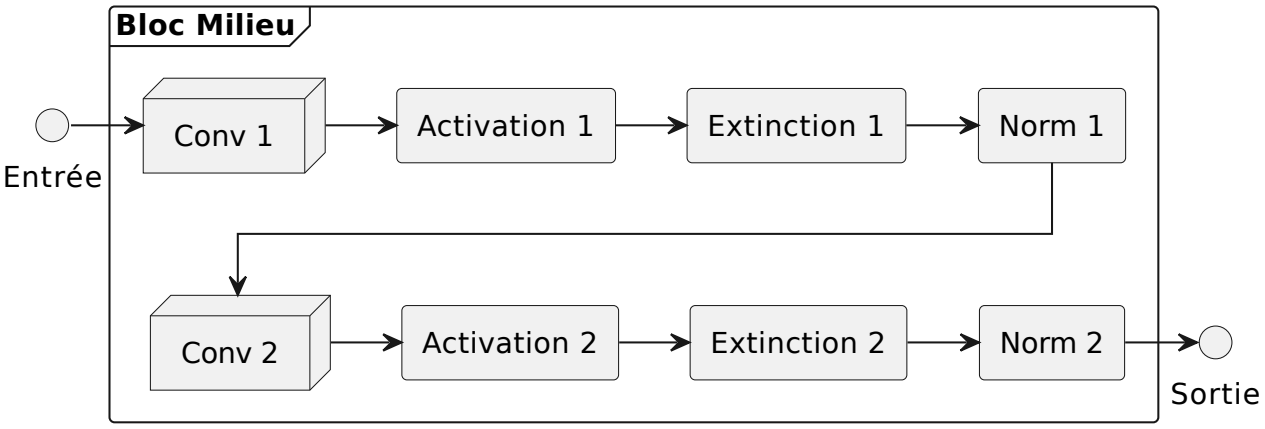


Figure A.3 – Bloc milieu pour notre U-Net sans **encodeur pré-entraîné VGG16**

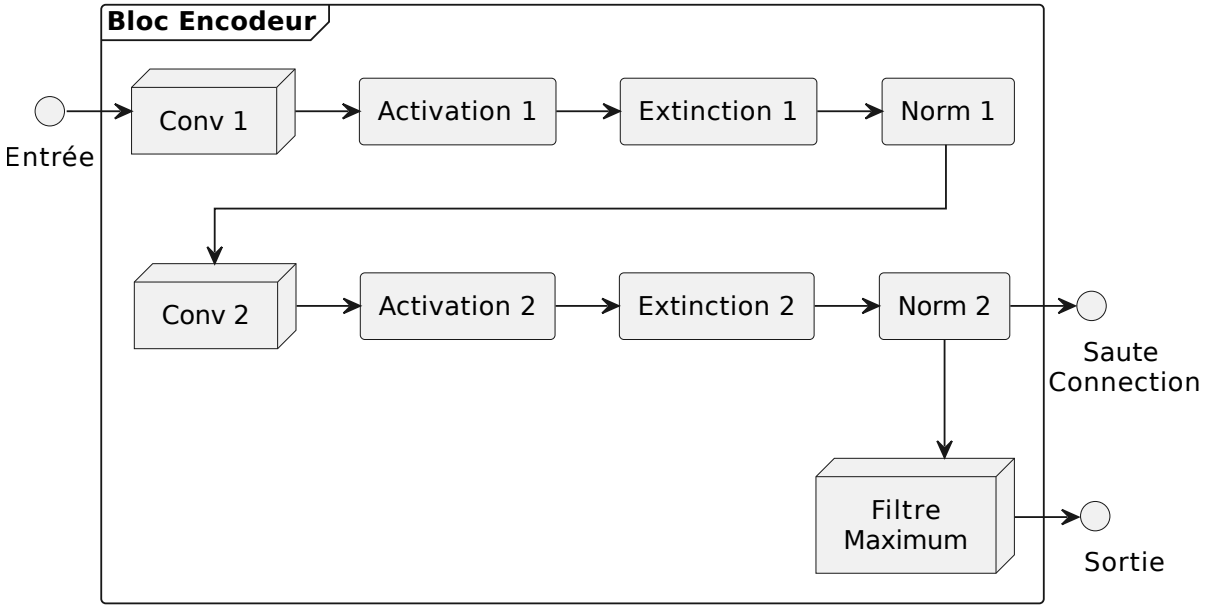


Figure A.4 – Bloc d'encodage pour notre U-Net sans **encodeur pré-entraîné VGG16**

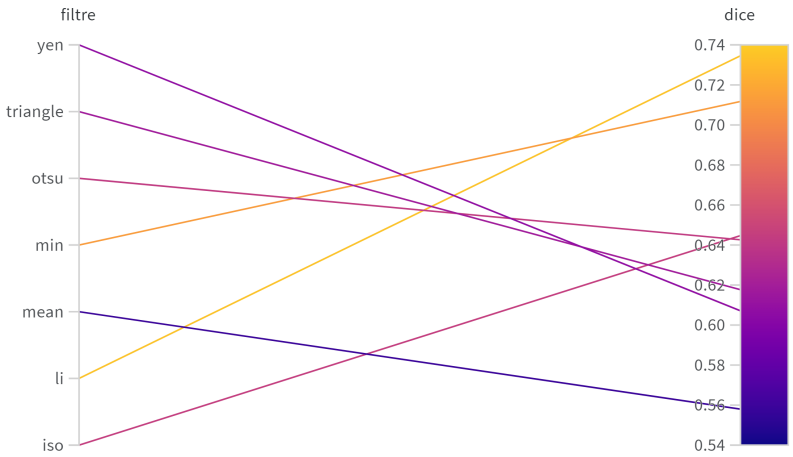


Figure A.5 – Coefficient de Dice moyen avec notre approche de segmentation par seuillage pour nos volumes annotés. *filtre* est le nom de la méthode de seuillage globale utilisée.

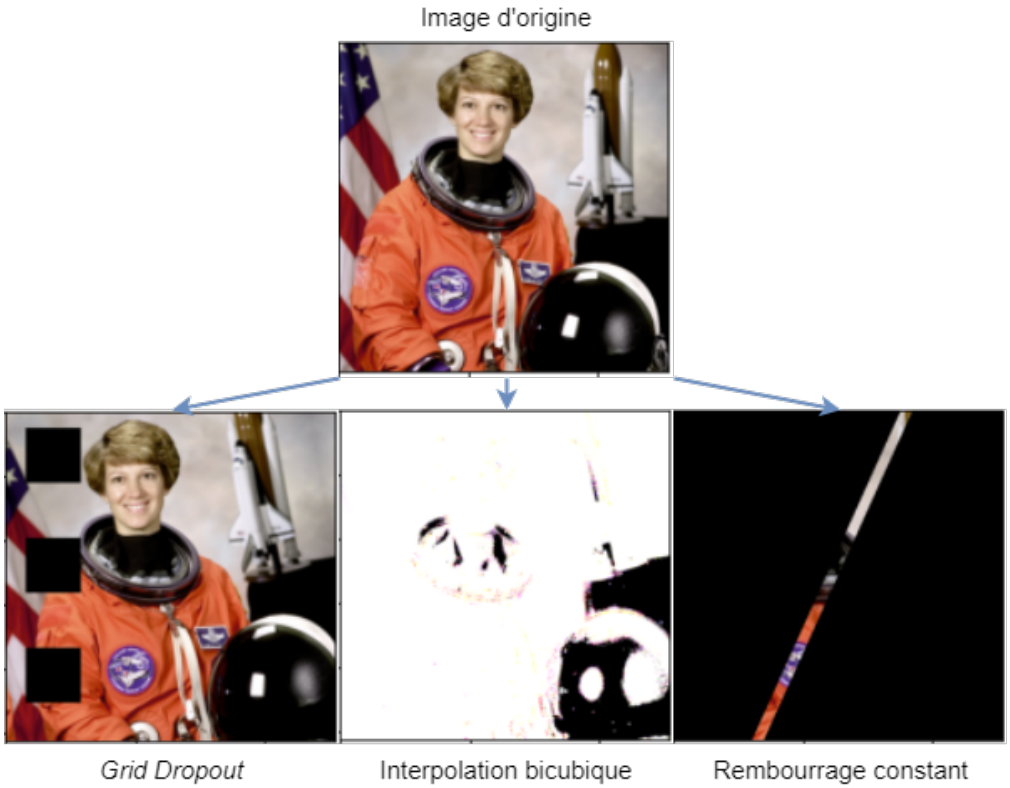


Figure A.6 – Exemples d'augmentation 3D problématiques avec le module Python Volumentations (Solovyev *et al.*, 2022). Le volume utilisé est constitué de 5 images superposées. La première transformation à gauche est *Grid Dropout* qui s'applique en diagonale sur la profondeur du volume, tandis que les régions retirées devraient être sur l'entièreté de la largeur, peu importe la profondeur. La transformation au centre est une rotation avec interpolation bicubique et on perd l'étendue des intensités d'origines, d'où la saturation des couleurs. La dernière transformation à droite est pour une rotation avec un rembourrage constant.

DETECTING THE 3D MORPHOLOGY OF CELL CYTONEMES USING SKELETONIZATION

Philippe Lemieux[⋆] *Basile Rambaud*[†] *Sébastien Carreno*[†] *Joël Lefebvre*[⋆]

[⋆]Laboratory of Digital Imaging, Neurophotonics and Microscopy, Université du Québec à Montréal, Canada
[†] Institute for Research in Immunology and Cancer, Université de Montréal, Canada

ABSTRACT

The characterization of three dimensional (3D) cell morphology is a complex task due the high variability in cell appearance and the lack of tools for 3D analysis. This results in time consuming visual analysis which obtains subjective results. In this paper, we explore the use of image processing methods such as deconvolution, labelling and skeletonization to automatize the collection of metrics about cell cytonemes in volumetric data. This pipeline revealed that skeletonizing an entire cell works well for bigger cytonemes, but still misses most of the shorter ones. This could inspire future works on the skeletonization of 3D objects such as cells to achieve more accurate results and offer new tools for research on cell morphology.

***Index Terms*—** Cytoneme, Cell morphology, Skeletonization, Deconvolution, Segmentation

1. INTRODUCTION

The characterization of 3D cell morphology is a complex task due to the high variability in cell appearances, the various imaging modalities that can be used, the complexity and high computational load originating from the 3D nature of the data, etc. Due to these challenges, many biologists still rely on visual inspections and labor-intensive manual methods to extract relevant cell morphology information. Thus, the development of dedicated image processing pipelines will provide helpful tools to researchers when evaluating 3D cell morphology.

For instance, in a collaborative project with the University of Montréal, we aim to identify the mechanisms controlling the biogenesis of cytonemes. These specialized signaling cell extensions are implicated in cell-cell communication during embryonic development [1]. Interestingly, cytonemes were also recently shown to contribute to tumor growth progression and malignancy [2]. Using confocal microscopy, cell expressing cytonemes can be observed and stored as 3 dimensional (3D) grayscale volumes. Currently, manual methods are used to characterize the cytonemes in these volumes. For example, two spheres are used to calculate the length of the furthest cytoneme for a given cell. Both spheres are centered visually on the cell. The first sphere radius extends to the furthest voxel from the center, and the second sphere extends to the start of the cytoneme associated with the previous voxel (**Fig.1**). The difference of radius is interpreted as the length of the longest cytoneme. This characterization process is performed by manually iterating over one axis of a volume, showing consecutive 2 dimensional (2D) image slices.

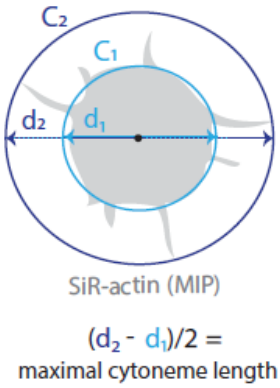


Fig. 1. Visual analysis method.

While this method might be efficient for quick analyses, this solution has some limitations. First, the tortuosity of the cytoneme is not considered, therefore it might not be the longest for a given cell. Second, most of this process is done by hand by the biologist, which is time-consuming and subjective. The automation of this process would help alleviate these issues by minimizing researchers' inputs and removing the need of slicing a volume for the sake of visualization and annotation.

In this paper, we present an image-processing method for cytoneme characterization. Specifically, we explore the use of skeletonization to tackle the study of cytoneme morphology in volumetric data. We have implemented a cell surface segmentation pipeline, followed by a skeletonization algorithm. The skeleton is then divided into different regions and explored using a graph theory approach similar to depth-first search. This will allow a precise and automated quantification

Corresponding author: lefebvre.joel@uqam.ca. More details may be shared on our laboratory's website : https://linum.info.uqam.ca

of the number, length and branching of cytonemes to characterize the mechanisms controlling their biogenesis.

2. METHODOLOGY

The objective of this project is to develop an automated method to extract cell cytonemes information from volumetric confocal data (**Fig.2(a)**). Some of the extracted information consist of the number of cytonemes per cell, their position, length and number of branches. A popular approach for this problem would have been to use a convolutional neural network, such as a U-Net [3], for the segmentation of specific cellular components and regions of interest (ROI). We opted for an image processing approach due to a lack of annotated data available for this problem and because creating a sizable dataset for training can be tedious. The method used can be summarized as: volume pre-processing, deconvolution, segmentation of cells, separation of connected cells and identification of cytonemes and their centerlines. Finally, cytonemes characteristics can be quantified once the cytonemes coordinates are identified starting from the cell body.

2.1. Cell preparation and imaging

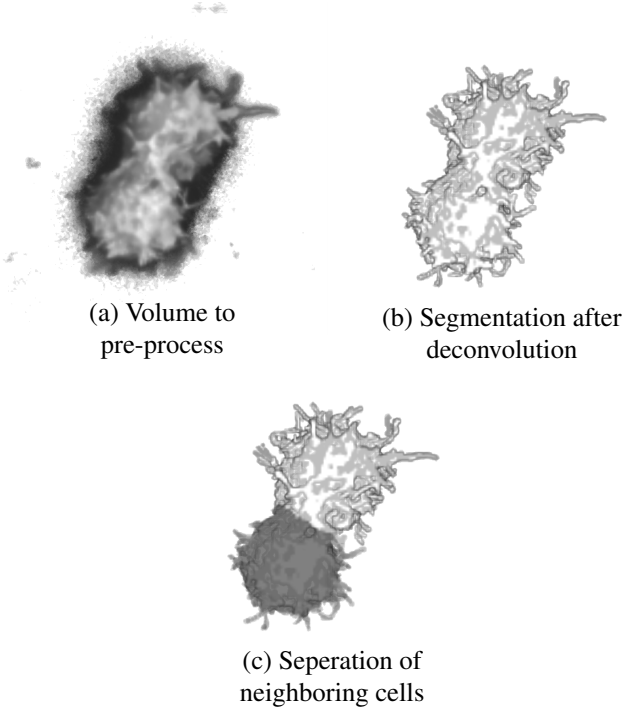
S2 cell culture and cDNA transfection: Drosophila S2 cells were grown at 25°C in Schneider's Drosophila medium (21720001; GIBCO) complemented with 1% Penicillin-Streptomycin antibiotics and 10% FBS (12483020; Invitrogen). cDNA transfection of EGFP tagged proteins was performed using FuGENE HD Reagent (Promega) diluted in Opti-MEM (Invitrogen) media. Cells were plated in 96 wells glass bottom plate for microscopy (82050-792; Sensoplate; Greiner-Bio-One) and transfected for 48h.

Image acquisition and annotation: Image acquisition was realized on live cells in Schneider's medium at room temperature with a Zeiss spinning disk confocal microscope Observer Z1 equiped with a 488 laser, a x63 objective (PlanApochromat DIC 1.4) and a Zeiss AxioCam 506 mono (typically of lateral resolution = 120nm and axial resolution = 270nm). These resulted in multiple volumetric stacks, each containing a few cells. Some cells were cropped and labeled manually by B.R. using the FIJI software (National Institute of Health)[4]. These annotated volumes were used to evaluate the image-processing pipeline performance during development.

2.2. Data pre-processing

Applying a global intensity threshold to the unprocessed cell's volume would result in keeping a lot of noises and failing to segment a cell in its entirety, the cytonemes in particular. For these reasons, volume pre-processing is performed (see **Fig.2(b)**). First, the volumes are deconvolved using the Richardson-Lucy method with 20 iterations. Afterwards, to obtain clean cytonemes segmentation, we sharpen the cell's edges with the maximums of 3 Laplacian of Gaussian filter (sigma 1.5, 2 and 4). The result is normalized with the threshold value and added to the volume. Cells are then segmented with the Otsu thresholding method[5]. Voxel intensity inside the cells nucleus tends to be low and the cell segmentation may contain holes. If these cavities are fully enclosed by the segmentation, they are filled using a 3D morphological hole filling algorithm. Then, small objects are considered as noise and removed, again with morphological filters. This enhancement technique was inspired by the pre-processing method used for the automatic detection of sub-cellular morphological motifs[6].

Once the cells surface is identified, cells need to be labelled individually, even if they are touching one another. To separate adjacent cells, we first calculate the Euclidean distance between every voxel inside the segmented cell to the closest background voxel. From this distance map, we select the local maxima and interpret them as the center of each adjacent cell. To avoid detecting local maxima that are too close to one another, a newly found local maximum within a distance of 2 microns from an already found one are not kept. The local maxima are then used as seed for the watershed method implemented by the scikit-image Python module[7] (**Fig.2(c)**). Watershed separates the cell which are then be labelled.



(a) Volume to pre-process

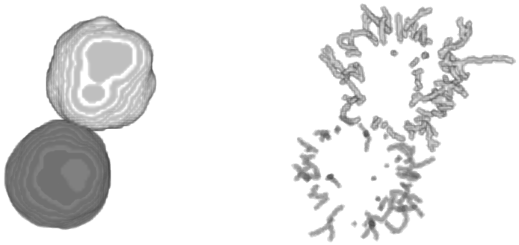
(b) Segmentation after deconvolution

(c) Seperation of neighboring cells

Fig. 2. Volumetric data pre-processing.

2.3. Cells segmentation and skeletonization

Hypothetically, skeletonizing an entire cell could not only clearly identify the path of each cytoneme, but also record their orientation starting from the cell body. Contrarily to only skeletonizing the cytonemes, this would avoid having to guess afterward the starting point of cytonemes and their directions. The skeletonization method we used is from Lee Ta-Chih[8] as implemented by the scikit-image Python module[7]. To only keep the skeleton of the cytonemes, cell body is estimated and is excluded from the final skeleton. The body is calculated with a morphological opening filter using a spherical structural element (radius = 2 microns) (**Fig.3(a)**), removing the cytonemes and leaving behind an good approximation of the cell body (**Fig.3(b)**). The intersection between the body edge and the skeleton is kept for the next step.



(a) Cell bodies

(b) Skeleton excluding bodies

Fig. 3. Skeletonization result after removal of the body section (skeleton dilated for visualisation).

2.4. Skeleton post-processing

At this point, every object in the 3D skeleton is a cytoneme. From here we want to find all the cytoneme paths, meaning all the voxels, from start to end, representing the skeletonized cytonemes. A cytoneme starting point is the intersection between a segmented object and the body of the cell previously found. Every voxel with only one neighbour that is not a start point is set as an end point of a cytoneme. Cytonemes branch in and out, in our case, cytonemes paths can use the same voxels except if it's an end point, in which case only the best cytoneme will be kept. To visit the cytonemes from start to end, we proceed with an approach similar to depth-first-search (DFS), with the exception that a voxel can be visited more than once unless it already exists in the current path. One path is allowed per end point. Paths sharing the same end point are filtered based on the consistency of orientation and shortest length. For each 3 consecutive voxels, the consistency of orientation is based on the smallest difference of dot products between the first two voxels and last two voxels. For example, let's consider the vector $\vec{A}$ representing the line between points 1 and 2 (**Fig. 4**), $\vec{B}$ for the line between 2 and 3, and $\vec{C}$ for the line between 2 and 4. If $\vec{A} \cdot \vec{B} < \vec{A} \cdot \vec{C}$, then the first path $1 \rightarrow 2 \rightarrow 3$ has a more consistent orientation than the path $1 \rightarrow 2 \rightarrow 4$ and will be chosen by the algorithm.

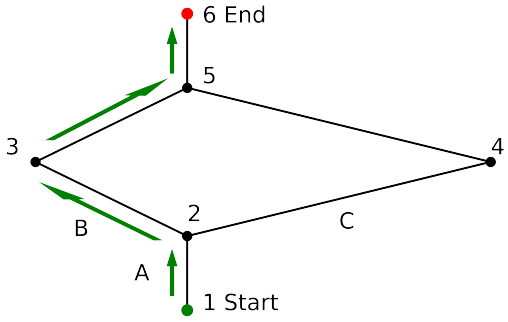


Fig. 4. The path found would be [1, 2, 3, 5, 6], [1, 2, 3 , 5, 4], [1, 2, 4, 5, 6] and [1, 2, 4, 5, 3]. Paths that do not end with 6 will be discarded as well as [1, 2, 4, 5, 6] since only 1 path per ending is allowed and [1, 2, 3, 5, 6] (green arrows) has a more consistent orientation.

3. RESULT AND DISCUSSION

3.1. Evaluation of pipeline

For the pipeline validation, B.R. prepared 2 kinds of annotations for 9 volumes, one for the cells' bodies and another for the cells' cytonemes. Using these, we validated our segmentation, body approximation and skeleton.

For validating the segmentation of the cells, we combined the cytonemes and bodies annotations and used the result to create a confusion matrix. We did the same thing for the cell body segmentation, using the corresponding cell body annotations. Our validation of the cells segmentation performed well with an average Dice coefficient of 0.79. The average Dice coefficient for the cell bodies was quite low at 0.5, but since it is only used to estimate the beginning of cytonemes, it doesn't impact the pipeline too much.

Finally, for validating the skeletonized cytonemes found after post-processing, we used the cytonemes annotations and a testing method based an evaluation protocol for skeletonization [9] to categorize voxels into 3 classes: true positive (TP), false positive (FP) and false negative (FN). The difference between our skeleton validation method and the evaluation protocol [9] is the buffer zone employed with the reference skeleton for validation differed since we already had precise annotations of cytoneme, removing the need to create this buffer ourselves. The buffer zone for the extracted skeleton for cytonemes has a width of 2 microns. Our validation revealed an average recall of 0.32 and an average precision of 0.27. This is mostly explained by how difficult smaller cytonemes are to detect with our approach, as well as the lack of distinction between the cytonemes and the noise caused by the lamellipodia

edges. Comparing our pipeline to the current visual method, the cytoneme length calculated is more accurate, taking into consideration the tortuosity. Our pipeline can at least automatically analyses cytonemes who are not too small for the skeletonization method used.

The microscopy volumes used to test the pipeline have varying dimension (typically around 45x290x314). For the 9 volumes containing 27 cells, the pipeline took 71 seconds on averages to complete. The computer used for this test has a AMD Ryzen 5 1600 6 cores processor and 16 GB of RAM. To optimize our pipeline, the DFS approach for exploring skeleton objects could start from the end point and favor branching with the closest orientation. As soon as a start point is reached, the whole process would end and there would not be any need for pruning paths sharing the same end point.

3.2. Skeletonization limitations

With our results, Lee skeletonization method does not reach the level of details we hoped for. Including the cell body in the skeletonization process results in more complex skeleton connectivity of the object and multiple smaller cytonemes are missed. It does not help that there is not any parameters to adjust the resulting skeleton. Another option that could be explored in future work would be to try a different type of skeletonization method in 3D, such as the medial axis or Voronoi algorithms.

On another note, if implementing 3D skeletonization methods proves to be too complex, what about assisted skeletonization? For example, let's consider the Voronoi skeletonization method. It is probably the easiest skeletonization to implement in 2D, but in 3D, ridges becomes polygons and interpreting the 3D skeleton becomes much harder. The resulting ridges could be used as pre-processing step to remove most of the cell body, which causes problems for the skeleton accuracy. Skeletonizing the polygon ridge would "guide" the skeletonization better preserve the cell and cytonemes connectivity.

4. CONCLUSION

To conclude, this initial cytoneme processing pipeline is less accurate than we hoped, missing most cell cytonemes. Even so, it should be faster and do a more in-depth analysis of cytonemes and their tortuosity. Skeletonizing the entirety of the cells is clearly lacking in accuracy for cytonemes who don't protrude enough from the cell body. Another issue is that while the cell seperation process seems simple, it hasn't been validated. Issues did occur at time when the cells were not spherical enough or that holes in the cells nucleus couldn't be filled. More annotations with varying labels for cells would be needed for validation.

For further analysis, future work could be explored with guided skeletonization or artificial intelligence to obtain more detailed skeletons of the entire cell or cytonemes. Especially the multiple volume annotations which could be reused as a base for convolutional model training.

5. ACKNOWLEDGMENTS

P.L. was supported by research scholarships originating from J.L.'s NSERC Discovery (RGPIN-2020-06109) and FRQNT Start-up (287523) grants. B.R. was supported by research scholarships originating from S.C.'s NSERC Discovery (RGPIN-2017-05170) and CIHR (PJT-162109) grants. The authors have no relevant financial or non-financial interests to disclose.

6. REFERENCES

[1] Chengting Zhang and Steffen Scholpp, "Cytonemes in development," *Current Opinion in Genetics & Development*, vol. 57, pp. 25–30, Aug. 2019.

[2] Sol Fereres et al., "Cytoneme-mediated signaling essential for tumorigenesis," *PLOS Genetics*, vol. 15, no. 9, pp. e1008415, Sept. 2019, Publisher: Public Library of Science.

[3] Olaf Ronneberger et al., "U-net: Convolutional networks for biomedical image segmentation," 2015, cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015.

[4] Johannes Schindelin et al., "Fiji: an open-source platform for biological-image analysis," *Nature Methods*, vol. 9, no. 7, pp. 676–682, Jul 2012.

[5] N. O., "10.1109/tsmc.1979.4310076," *IEEE*, vol. 9, no. 1, pp. 62–66, 1979.

[6] M. D. et al., "10.1038/s41592-019-0539-z," *Nat. Met.*, vol. 16, 10 2019.

[7] S. W. et al., "10.7717/peerj.453," *PeerJ*, vol. 2, pp. e453, 6 2014.

[8] T. L. et al., "10.1006/cgip.1994.1042," *CVGIP: Graph. Models Image Process.*, vol. 56, no. 6, pp. 462–478, Nov. 1994.

[9] R. Y. et al., "10.1109/dicta.2015.7371256," 2015, pp. 1–6.

INDEX

élément structurant Matrice utilisé pour appliquer une opération morphologique. 33, 48, 52

annotation douce Annotation composée de valeurs réelles au lieu de discrètes afin d'exprimer l'ambiguïté des classes.. 20

annotation binaire Annotation composée de valeurs discrètes pour identifier une classe.. 20

carte de caractéristiques Structure contenant le résultat des convolutions avec les filtres d'une couche de convolution. Traduit de l'anglais pour *features map* (DataFranca, 2022).. vi, 15, 37

centroïde Désigne le point de gravité d'une zone ou région. 44

connexion saute-couche Type de couche permettant de connecter une couche à l'autre sans respecter la séquence des couches du modèle. Traduit de l'anglais pour *skip connection* (DataFranca, 2022).. 13, 37, 56

couche entièrement connectée Traduit de l'anglais pour *fully connected layer*(DataFranca, 2022). 37

doublon Réutilisation d'une entrée dans un lot lors de l'entraînement.. 38

encodeur pré-entraîné Un modèle encodeur préférablement pré-entraîné qui remplace l'encodeur d'un U-Net. Traduit de l'anglais pour *backbone*.. viii, 15, 17, 33, 36, 37, 40, 77

extinction de neurone Opération qui désactive un pourcentage des caractéristiques d'une couche afin d'éviter le surajustement. Traduit de l'anglais pour *dropout* (DataFranca, 2022).. vii, 17, 38, 39, 57, 58

extra-expert Entre les experts d'un domaine.. 1, 26

filtre maximum Filtre non-linéaire qui conserve la valeur maximum du voisinnage de chaque coordonnée.. 18

hyperparamètre Les hyperparamètres d'un modèle contrôle le processus d'entraînement. Par exemple, le taux d'apprentissage.. vii, ix, 15, 38, 45, 46, 47, 48, 53, 54, 59, 60, 70

image numérique Image avec des axes discrets, où chaque axe peut être représenté sous forme de liste.. 6

jeton caractéristique Sous-sections d'une image de dimensions similaire pouvant contenir de la métadata comme ça position. Traduction de l'anglais pour *token* dans les ViT.. 12

lamellipode Large extension membranaire autour du corps de cellule pour se déplacer.. 4, 33

masque géodésique Une contrainte pour une ou des régions prédéfinies dans une image.. 9

morphologique Lié à la forme.. 2

moyenne non-locale Technique classique de débruitage à l'état de l'art pour le 2D. Ce filtre remplace les pixels en fonction des régions qui y sont proches si elles ont un voisinnage similaire. Traduit de l'anglais pour *non-local means*. 26

mécasnisme d'attention Mécasnisme qui a la capacité d'apprendre à se concentrer sur des parties spécifiques d'une données complexes (DataFranca, 2022).. 12

noyau Filtre utilisé pour appliquer un filtre. Traduit de l'anglais pour *kernel*. 6, 12, 52, 60, 66

pas de gradient Réfère au traitement d'un lot lors d'une époche. Traduit de l'anglais pour *step*(DataFranca, 2022). 56

poids d'instance Poids d'entraînement spécifique pour chaque position d'une donnée d'entrée. Traduit de l'anglais pour *sample weight*.. 55

reconstruction d'un lot Traduit de l'anglais pour *inpainting*(DataFranca, 2022). 16

recoupement Traduit de l'anglais pour *overlap*(DataFranca, 2022). 41, 42, 60

recuite cosinus Méthode de modification du taux d'apprentissage lors de l'entraînement en fonction de l'époque. Traduit de l'anglais pour *cosine annealing*. 39, 54, 57, 58, 60

rembourrage Valeurs ajoutés à un ou plusiers axes pour les alongir. Traduit de l'anglais pour *padding*. 7, 8, 35, 37, 38, 40, 41, 53, 54

ré-échantillonnage avec remise ensembliste Combinaison de prédictions pour un seul résultat afin de réduire leur variance entre elles. Traduit de l'anglais pour *bagging*.. 66

réseau antagoniste génératif Traduit de l'anglais pour *Generative Adversarial Network* (DataFranca, 2022).. 17

sous-échantillonnage par valeur maximale Filtre non-linéaire qui conserve la valeur maximale avec son noyau pour chaque position dans l'image. Traduit de l'anglais pour *max pooling*(DataFranca, 2022). 34, 37

tortuosité Lié aux courbures, soit la sinuosité.. 3

transformeur visuel Traduction de l'anglais pour *Vision Transformer* (ViT).. 12, 66

VGG16 Réseau de neurones avec 16 couches convolutionnelles. Spécialisé pour des tâches de classifications.. vii, viii, 4, 15, 33, 36, 37, 56, 60, 77

voxel Même principe qu'un pixel dans une image numérisé, mais cette fois ci en trois dimensions.. xiii, 1, 10, 12, 18, 23, 26, 29, 31, 33, 36, 42, 44, 45, 50, 51, 52, 53, 55, 57, 60, 61, 62, 63, 64, 65, 66, 67, 70, 72

BIBLIOGRAPHIE

Alokasi, H. et Ahmad, M. B. (2022). The accuracy performance of semantic segmentation network with different backbones. *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*, 49–54.

Berg, S., Kutra, D., Kroeger, T., Straehle, C. N., Kausler, B. X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M., Eren, K., Cervantes, J. I., Xu, B., Beuttenmueller, F., Wolny, A., Zhang, C., Koethe, U., Hamprecht, F. A. et Kreshuk, A. (2019). ilastik : interactive machine learning for (bio)image analysis. *Nature Methods*. https://doi.org/10.1038/s41592-019-0582-9

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com, https://www.wandb.com/

BioImage.IO (2023). Model zoo bioimage.io. https://bioimage.io/

Burger, W. et Burge, M. (2010). *Principles of Digital Image Processing : Fundamental Techniques*. Undergraduate Topics in Computer Science. Springer London. https://books.google.ca/books?id=2LIMMD9FVXkC

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M. et Kalinin, A. A. (2020). Albumentations : Fast and flexible image augmentations. *Information*, *11*(2), 125. https://doi.org/10.3390/info11020125

Caicedo, J. C., Goodman, A., Karhohs, K. W., Cimini, B. A., Ackerman, J., Haghighi, M., Heng, C., Becker, T., Doan, M., McQuin, C., Rohban, M. H., Singh, S. et Carpenter, A. E. (2019). Nucleus segmentation across imaging experiments : the 2018 data science bowl. *Nature Methods*, *16*, 1247 – 1253.

Cardoso, M. J., Li, W., Brown, R., Ma, N., Kerfoot, E., Wang, Y., Murrey, B., Myronenko, A., Zhao, C., Yang, D., Nath, V., He, Y., Xu, Z., Hatamizadeh, A., Myronenko, A., Zhu, W., Liu, Y., Zheng, M., Tang, Y., … Feng, A. (2022). Monai : An open-source framework for deep learning in healthcare.

Chen, P., Liu, S., Zhao, H. et Jia, J. (2020). Gridmask data augmentation. *ArXiv*, *abs/2001.04086*.

Chollet, F. *et al.* (2015). Keras. https://github.com/keras-team/keras

Ciçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T. et Ronneberger, O. (2016). 3d u-net : Learning dense volumetric segmentation from sparse annotation. Dans *International Conference on Medical Image Computing and Computer-Assisted Intervention*.

Coll, B. et Morel, J.-M. (2011). Non-local means denoising. *Image Processing On Line*, *1*. https://doi.org/10.5201/ipol.2011.bcm_nlm

Czymmek, K. J., Whallon, J. H. et Klomparens, K. L. (1994). Confocal microscopy in mycological research. *Experimental Mycology*, *18*(4), 275–293. https://doi.org/https://doi.org/10.1016/S0147-5975(06)80001-0

DataFranca. (2022). *Les 101 mots de l'intelligence artificielle : petit guide du vocabulaire essentiel de la*

science des données et de l'intelligence artificielle. (première édition. éd.). DataFranca.org.

de Chaumont, F., Dallongeville, S., Chenouard, N., Hervé, N., Pop, S., Provoost, T., Meas-Yedid, V., Pankajakshan, P., Lecomte, T., Le Montagner, Y. et et al. (2012). Icy : An open bioimage informatics platform for extended reproducible research. *Nature Methods*, *9*(7), 690–696. https://doi.org/10.1038/nmeth.2075

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. et Fei-Fei, L. (2009). Imagenet : A large-scale hierarchical image database. Dans *2009 IEEE conference on computer vision and pattern recognition*, (p. 248–255). Ieee.

Ding, X., Zhang, X., Zhou, Y., Han, J., Ding, G. et Sun, J. (2022). Scaling up your kernels to 31×31 : Revisiting large kernel design in cnns. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11953–11965. https://api.semanticscholar.org/CorpusID:247446771

Dominik Kutra, EMBL Heidelberg ; Anna Kreshuk, E. H. (2022). interactive image segmentation with ilastik. https://youtu.be/F6KbJ487iiU?t=1491

Driscoll, M., Welf, E., Jamieson, A., Dean, K., Isogai, T., Fiolka, R. et Danuser, G. (2019). Robust and automated detection of subcellular morphological motifs in 3d microscopy images. *Nature Methods*, *16*. https://doi.org/10.1038/s41592-019-0539-z

Fereres, S. *et al.* (2019). Cytoneme-mediated signaling essential for tumorigenesis. *PLOS Genetics*, *15*(9), e1008415. Publisher : Public Library of Science, https://doi.org/10.1371/journal.pgen.1008415

Gani, H., Naseer, M. et Yaqub, M. (2022). How to train vision transformer on small-scale datasets ? Dans *British Machine Vision Conference*.

Glasbey, C. (1993). An analysis of histogram-based thresholding algorithms. *CVGIP : Graphical Models and Image Processing*, *55*(6), 532–537. https://doi.org/https://doi.org/10.1006/cgip.1993.1040

Gonzalez, R. et Woods, R. (2018). *Digital Image Processing*. Pearson. https://books.google.ca/books?id=0F05vgAACAAJ

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. et Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. et Schmidhuber, J. (2015). Lstm : A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*, 2222–2232.

Gros, C., Lemay, A. et Cohen-Adad, J. (2020). Softseg : Advantages of soft versus binary training for image segmentation. *Medical image analysis*, *71*, 102038.

Haeberlé, O., Bicha, F., Simler, C., Dieterlen, A., Xu, C., Colicchio, B., Jacquey, S. et Gramain, M. P. (2001). Identification of acquisition parameters from the point spread function of a fluorescence microscope. *Optics Communications*, *196*, 109–117. https://api.semanticscholar.org/CorpusID:31680708

He, K., Zhang, X., Ren, S. et Sun, J. (2015). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hörl, D., Rusak, F. R., Preusser, F., Tillberg, P., Randel, N., Chhetri, R. K., Cardona, A., Keller, P. J., Harz, H., Leonhardt, H., Treier, M. et Preibisch, S. (2018). Bigstitcher : Reconstructing high-resolution image datasets of cleared and expanded samples. *bioRxiv*. https://doi.org/10.1101/343954

Hughes, C. (2022). Transfer learning on greyscale images : How to fine-tune pretrained models on black-and-white... http://tinyurl.com/sy53cv93

Indolia, S., Goswami, A. K., Mishra, S. et Asopa, P. (2018). Conceptual understanding of convolutional neural network- a deep learning approach. *Procedia Computer Science*, *132*, 679–688. International Conference on Computational Intelligence and Data Science, https://doi.org/https://doi.org/10.1016/j.procs.2018.05.069

Ioffe, S. et Szegedy, C. (2015). Batch normalization : Accelerating deep network training by reducing internal covariate shift. *ArXiv*, *abs/1502.03167*.

Isensee, F., Petersen, J., Klein, A., Zimmerer, D., Jaeger, P. F., Kohl, S. A. A., Wasserthal, J., Koehler, G., Norajitra, T., Wirkert, S. J. et Maier-Hein, K. (2018). nnu-net : Self-adapting framework for u-net-based medical image segmentation. *ArXiv*, *abs/1809.10486*.

Kekre, S. *et al.* (2023). Streamlit. https://github.com/streamlit.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. et Tang, P. T. P. (2016). On large-batch training for deep learning : Generalization gap and sharp minima. *CoRR*, *abs/1609.04836*. http://arxiv.org/abs/1609.04836

Khandelwal, V. (2020). The architecture and implementation of vgg-16. https://pub.towardsai.net/the-architecture-and-implementation-of-vgg-16-b050e5a5920b

Kramm, B., Rivera, N., Hernández, C. et Baier, J. A. (2021). A suboptimality bound for 2k grid path planning. Dans *Symposium on Combinatorial Search*.

L., T. *et al.* (1994). 10.1006/cgip.1994.1042. *CVGIP : Graph. Models Image Process.*, *56*(6), 462–478. https://doi.org/10.1006/cgip.1994.1042

Lee, H.-Y., Huang, J.-B., Singh, M. K. et Yang, M.-H. (2017). Unsupervised representation learning by sorting sequences. *2017 IEEE International Conference on Computer Vision (ICCV)*, 667–676.

Lemieux, P. (2023). Carreno cytonemes. https://github.com/linum-uqam/carreno-cytonemes.

Lemieux, P., Rambaud, B., Carreno, S. et Lefebvre, J. (2022). Detecting the 3d morphology of cell cytonemes using skeletonization. *IEEE 19th International Symposium on Biomedical Imaging (ISBI)*.

Li, C. et Lee, C. (1993). Minimum cross entropy thresholding. *Pattern Recognition*, *26*(4), 617–625. https://doi.org/https://doi.org/10.1016/0031-3203(93)90115-D

Loshchilov, I. et Hutter, F. (2016). Sgdr : Stochastic gradient descent with warm restarts. *arXiv : Learning*.

https://api.semanticscholar.org/CorpusID:14337532

Lundh, F. (1999). An introduction to tkinter. https://api.semanticscholar.org/CorpusID:59755411

Mo, Y., Wu, Y., Yang, X., Liu, F. et Liao, Y. (2022). Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, *493*, 626–646.

Mustafa, B., Loh, A., von Freyberg, J., MacWilliams, P., Wilson, M., McKinney, S. M., Sieniek, M., Winkens, J., Liu, Y., Bui, P., Prabhakara, S., Telang, U., Karthikesalingam, A., Houlsby, N. et Natarajan, V. (2021). Supervised transfer learning at scale for medical imaging. *ArXiv*, *abs/2101.05913*.

Müller, D. et Kramer, F. (2021). Miscnn : a framework for medical image segmentation with convolutional neural networks and deep learning. *BMC Medical Imaging*, *21*. https://doi.org/10.1186/s12880-020-00543-7

Neubert, P. et Protzel, P. (2014). Compact watershed and preemptive slic : On improving trade-offs of superpixel segmentation algorithms. Dans *2014 22nd International Conference on Pattern Recognition*, (p. 996–1001). https://doi.org/10.1109/ICPR.2014.181

Newell, A. et Deng, J. (2020). How useful is self-supervised pretraining for visual tasks ? *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7343–7352.

Niblack, W. (1986). *An Introduction to Digital Image Processing*. Delaware Symposia on Language Studies5. Prentice-Hall International. https://books.google.ca/books?id=XOxRAAAAMAAJ

Nisa, S. Q. et Ismail, A. R. (2022). Dual u-net with resnet encoder for segmentation of medical images. *International Journal of Advanced Computer Science and Applications*.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62–66. https://doi.org/10.1109/TSMC.1979.4310076

Paetzold, J. (2021). Jocpae/cldice. https://github.com/jocpae/clDice

Paetzold, J. C., Shit, S., Ezhov, I., Tetteh, G., Helmholtz, A. E., Munich, Z. et Menze, B. H. (2019). cldice - a novel connectivity-preserving loss function for vessel segmentation. Dans *N/A*. https://api.semanticscholar.org/CorpusID:221081495

Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T. et Efros, A. A. (2016a). Context encoders : Feature learning by inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2536–2544.

Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T. et Efros, A. A. (2016b). Context encoders : Feature learning by inpainting. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2536–2544.

Pérez-García, F., Sparks, R. et Ourselin, S. (2021). Torchio : A python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *Computer Methods and Programs in Biomedicine*, *208*, 106236. https://doi.org/10.1016/j.cmpb.2021.106236

Qasim, I. (2021). A review on image segmentation techniques and its recent applications. *International Journal of Computer Science and Mobile Computing*.

Raghu, M., Zhang, C., Kleinberg, J. M. et Bengio, S. (2019). Transfusion : Understanding transfer learning for medical imaging. Dans *Neural Information Processing Systems*.

Richardson, W. H. (1972). Bayesian-based iterative method of image restoration∗. *J. Opt. Soc. Am.*, *62*(1), 55–59. https://doi.org/10.1364/JOSA.62.000055

Ridler, T. et Calvard, S. (1978). Picture thresholding using an iterative selection method. *IEEE Transactions on Systems, Man, and Cybernetics*, *8*(8), 630–632. https://doi.org/10.1109/TSMC.1978.4310039

Ronneberger, O., Fischer, P. et Brox, T. (2015). U-net : Convolutional networks for biomedical image segmentation. *CoRR*, *abs/1505.04597*. http://arxiv.org/abs/1505.04597

Ruan, B., Shuai, H.-H. et Cheng, W.-H. (2022). Vision transformers : State of the art and research challenges. *ArXiv*, *abs/2207.03041*.

Rueden, C. T., Schindelin, J., Hiner, M. C., DeZonia, B. E., Walter, A. E., Arena, E. T. et Eliceiri, K. W. (2017). Imagej2 : Imagej for the next generation of scientific image data. *BMC Bioinformatics*, *18*(1). https://doi.org/10.1186/s12859-017-1934-z

Sage, D., Donati, L., Soulez, F., Fortun, D., Schmit, G., Seitz, A., Guiet, R., Vonesch, C. et Unser, M. A. (2017). Deconvolutionlab2 : An open-source software for deconvolution microscopy. *Methods*, *115*, 28–41. https://api.semanticscholar.org/CorpusID:15348122

Sahoo, S. (2022). 2d convolution using python & numpy. https://medium.com/analytics-vidhya/2d-convolution-using-python-numpy-43442ff5f381

Sauvola, J. et Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern Recognition*, *33*(2), 225–236. https://doi.org/https://doi.org/10.1016/S0031-3203(99)00055-2

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J.-Y., White, D. J., Hartenstein, V., Eliceiri, K., Tomancak, P. et Cardona, A. (2012). Fiji : an open-source platform for biological-image analysis. *Nat Meth*, *9*(7), 676–682. http://dx.doi.org/10.1038/nmeth.2019

Schmid, B., Schindelin, J. E., Cardona, A., Longair, M. et Heisenberg, M. (2010). A high-level 3d visualization api for java and imagej. *BMC Bioinformatics*, *11*, 274 – 274.

Schmidt, J. (2023). Creating and training a u-net model with pytorch for 2d & 3d semantic segmentation : Model building... http://tinyurl.com/58jjmf3d

Shit, S., Paetzold, J. C., Sekuboyina, A. K., Ezhov, I., Unger, A., Zhylka, A., Pluim, J. P. W., Bauer, U. et Menze, B. H. (2020). cldice - a novel topology-preserving loss function for tubular structure segmentation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16555–16564.

Siddique, N. A., Paheding, S., Elkin, C. P. et Devabhaktuni, V. K. (2020). U-net and its variants for medical

image segmentation : A review of theory and applications. *IEEE Access*, *9*, 82031–82057.

Simonyan, K. et Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1556*.

Smith, L. N. (2017). Cyclical learning rates for training neural networks.

Smith, L. N. et Topin, N. (2018). Super-convergence : Very fast training of neural networks using large learning rates.

Sofroniew, N., Evans, K., Nunez-Iglesias, J., Solak, A. C., Lambert, T., kevinyamauchi, Freeman, J., Royer, L., Axelrod, S., Boone, P., Tung, T., jakirkham, Vemuri, P., Huang, M., Har-Gil, H., Buckley, G., Bryant, Rokem, A., wconnell, ... de Siqueira, A. (2020). *napari : 0.2.9* (v0.2.9) [Logiciel]. https ://doi.org/10.5281/zenodo.3603551

Solovyev, R., Kalinin, A. A. et Gabruseva, T. (2022). 3d convolutional neural networks for stalled brain capillary detection. *Computers in Biology and Medicine*, *141*, 105089. https://doi.org/10.1016/j.compbiomed.2021.105089

Song, Y., Dhariwal, P., Chen, M. et Sutskever, I. (2023). Consistency models. *arXiv preprint arXiv :2303.01469*.

W., S. *et al.* (2014). 10.7717/peerj.453. *PeerJ*, *2*, e453. https://doi.org/10.7717/peerj.453

W., S. *et al.* (2023a). Skeletonize. https://scikit-image.org/docs/stable/auto_examples/edges/plot_skeleton.html

W., S. *et al.* (2023b). Watershed segmentation. https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_watershed.html

Wang, X., Bo, L. et Li, F. (2019). Adaptive wing loss for robust face alignment via heatmap regression. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 6970–6980.

Warnes, Z. (2023). Hyperparameter tuning - always tune your models. https://towardsdatascience.com/hyperparameter-tuning-always-tune-your-models-7db7aeaf47e9

Yen, J.-C., Chang, F.-J. et Chang, S. (1995). A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing*, *4*(3), 370–378. https://doi.org/10.1109/83.366472

Yu, P., Sun, C. et Sun, M. (2022). Data efficient 3d learner via knowledge transferred from 2d model. Dans *European Conference on Computer Vision*.

Zeiler, M. D. et Fergus, R. (2013). Visualizing and understanding convolutional networks. *ArXiv*, *abs/1311.2901*. https://api.semanticscholar.org/CorpusID:3960646

Zhang, C. et Scholpp, S. (2019). Cytonemes in development. *Current Opinion in Genetics & Development*, *57*, 25–30. https://doi.org/10.1016/j.gde.2019.06.005

Zhang, R., Du, L., Xiao, Q. et Liu, J. (2020). Comparison of backbones for semantic segmentation network. *Journal of Physics : Conference Series*, *1544*.

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N. et Liang, J. (2018). Unet++ : A nested u-net architecture for medical image segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support : 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain, S...*, *11045*, 3–11.