Université LAVAL

Apprentissage et reconnaissance GIF-4101/GIF-7005

Projet : Détection automatique de prolongements neuronaux

Auteurs :
Baptiste Amato
Alexandre Chavenon
Arnoud Vanhuele

1 Introduction

Présentation du projet

Le projet est proposé par le centre de recherche CERVO. Il consiste à reconnaître des axones et des dendrites sur des images d'une protéine (actine), en étiquetant ces images n'ayant pas de marqueurs axonaux et dendritiques. Nous disposons d'une banque de données d'images déjà marquées : il s'agit donc un problème d'apprentissage supervisé.

Jeu de données

Le jeu de données initial comprend 1024 images au format *.tiff*, ayant chacune 3 canaux : un pour l'actine (la protéine d'intérêt), un pour les axones, et un pour les dendrites.
Ce jeu de données étant relativement petit pour un apprentissage par réseau neuronal, nous allons utiliser des méthodes d'augmentation comme les symétries, rotations, ou encore découpes de sous-parties des images.

Etat de l'art

Il s'agit ici de détecter différents objets dans une image (axones et dendrites à partir d'une image globale d'actine) : c'est un problème de détection particulier, car il n'est pas possible d'encadrer les objets par des "bounding boxes", utilisées par exemple pour la détection de visage, de personnes ou de voitures ; on cherche alors à détecter le contour des objets. Un article de recherche assez récent a démontré une capacité de détection de contour impressionnante : *Object Contour Detection with a Fully Convolutional Encoder-Decoder Network*, par **Yang *et al.***. Nous pensons donc nous orienter vers un réseau de neurones profond avec une architecture *Encoder-Decoder* ; cette architecture est aussi utilisé dans les traductions de textes (séquences en entrée et sortie). Des résultats probants concernant de la segmentation d'images sont présentés dans l'article *Iterative Deep Convolutional Encoder-Decoder Network for Medical Image Segmentation*, par **Jung Uk Kim, Hak Gu Kim, et Yong Man Ro**, suivant une architecture similaire (*Encoder-Decoder*).
Les principes de segmentation d'image sont clairement expliqués dans l'article **Fully Convolutional Networks for Semantic Segmentation**, par **Shelhamer *et al.***.
Concernant l'augmentation de notre jeu de données, nous aurons une approche classique par traitement d'image traditionnel et dans un second temps, nous explorerons la possibilité d'utiliser un Generative Adversarial Network pour l'augmentation des données, comme précisé dans l'article*Biomedical Data Augmentation Using Generative Adversarial Neural Networks. Artificial Neural Networks and Machine Learning* par **Calimeri, F. *et al.***.

2 Pré-traitements

Masques

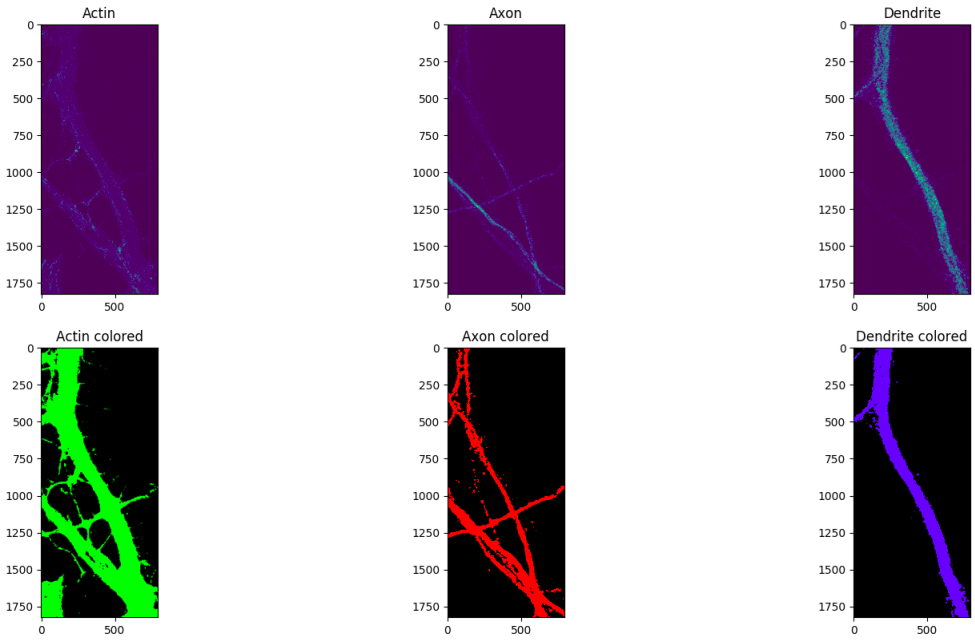

Figure 2.1: En haut, un triplet actine-axone-dendrite, et en bas les mêmes images une fois le masque appliqué

TODO: expliquer comment sont faits les masques

Taille de images d'entrée

Les images en entrée sont toutes de dimensions différentes, mais de même résolution. Il est donc nécessaire de conserver ces résolutions (ayant un sens physique) en n'effectuant aucun redimmensionnement. Le problème est que le réseau de neurones attend des images de même taille en entrée : nous avons donc découpé chacune des images en *crops* (petits carrés) de taille 224x224, correspondant à la taille des images en entrée du réseau **VGG16**, mentionné plus bas.

Prédiction d'une nouvelle image

Là encore, il est nécessaire de procéder par *crops* pour traiter l'image à traiter. Ainsi, chaque *crop* sera associé à une image de prédiction, et l'image résultante sera reconstituée à partir de ces *crops*.

3 Réseau de neurones

Nous utilisons la librairie *Keras* avec un back-end en *TensorFlow*. L'architecture sera celle présentée dans l'article de recherche *Object Contour Detection with a Fully Convolutional Encoder-Decoder Network*, par **Yang *et al.*** : la première parte du réseau est l'encodeur, basé sur l'architecture du réseau **VGG16**, en s'arrêtant juste avant le *Fully Connected layer*. Ensuite, le décodeur est celui décrit par l'article, permettant de reconstituer une image de la taille d'origine avec des opérations de *déconvolution*, qui en *Keras* se font grâce à l'opération *UpSampling* suivie d'une *same convolution* (convolution ne modifiant pas la taille de l'image d'entrée grâce à l'ajout de *padding*). + TODO: loss function + TODO: optimizer