Segmentation d'images

Contenu

[Introduction 2](#_Toc505269781)

[Présentation du Projet 2](#_Toc505269782)

[Objectifs 2](#_Toc505269783)

[Evaluation 2](#_Toc505269784)

[Exploration 2](#_Toc505269785)

[Type d'images 2](#_Toc505269786)

[Taille des images 3](#_Toc505269787)

[Couleur moyenne 3](#_Toc505269788)

[Pré-processing 4](#_Toc505269789)

[Préparation des masques 4](#_Toc505269790)

[Augmentation de Contraste 4](#_Toc505269791)

[Adaptative Threshold 4](#_Toc505269792)

[Redimensionnement 5](#_Toc505269793)

[Version non déformée 5](#_Toc505269794)

[Version classique 6](#_Toc505269795)

[Modèles 6](#_Toc505269796)

[Modèle Simple 6](#_Toc505269797)

[U-net Classique 6](#_Toc505269798)

[U-net Extended (ou multi-arm) 6](#_Toc505269799)

[Post-processing 6](#_Toc505269800)

[Analyse des résultats 6](#_Toc505269801)

[Pistes d'évolutions 6](#_Toc505269802)

[Conclusion 6](#_Toc505269803)

# Introduction

Dans l'objectif de faciliter le comptage et la détection de cellules par exemple dans le sang, il nous est demandé sur cette compétition Kaggle de faire une prédiction de la position de chaque cellule présentent sur une image. Nous allons donc explorer une nouvelle partie de la data analyse qu'est la **segmentation d'images.**

Pour ce faire, nous allons aborder le pré-processing, la modélisation avec 3 différents modèles puis le post-processing. En effet, comparé aux précédents projets, une phase importante de ce projet se situe sur la phase de post-processing. Pour finir, nous parlerons des résultats obtenus, les possibilités d'améliorations puis une conclusion.

# Présentation du Projet

## Objectifs

Comme évoqué dans l'introduction, nous devons prédire un masque pour chaque cellule de l'image. Pour l'entrainement, nous avons accès à 670 images avec pour chacune d'elle, n-images représentant le masque de chaque cellule. A partir de celles-ci, nous devons entrainer un modèle permettant de sortir n-mask couvrant au mieux chaque cellule. Concernant le test, on a 65 images.

## Evaluation

L'évaluation Kaggle sur ce projet un peu complexe. Dans un 1er temps, Kaggle vérifie qu'il n'y a aucune superposition de masques, sinon le score final vaut 0.

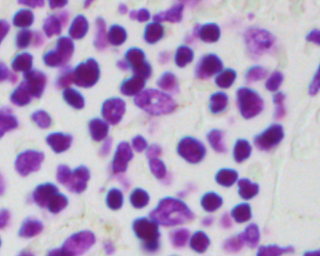
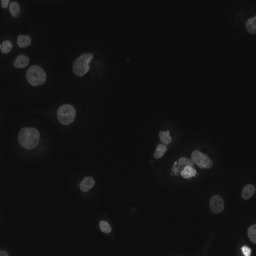
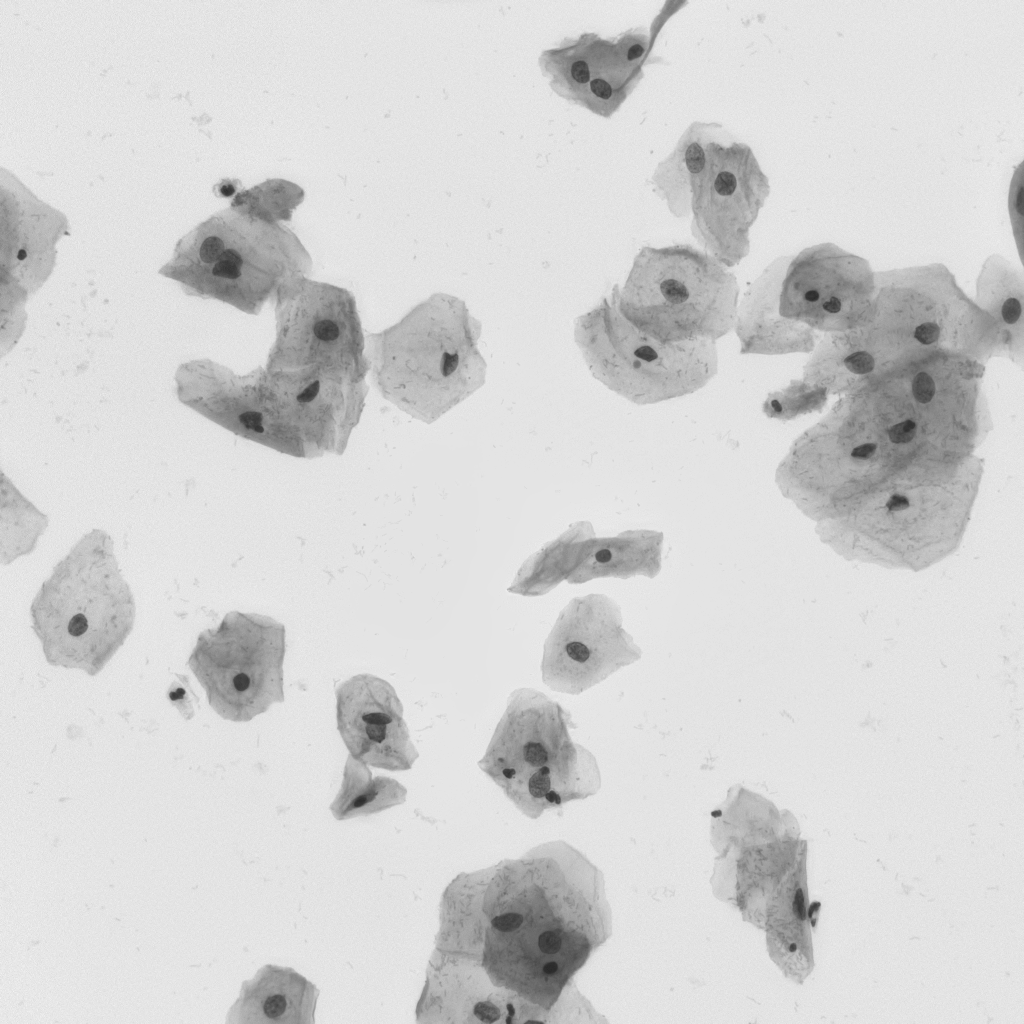
Ensuite, pour chaque image du test set, n-masques sont fournis. Ils sont comparés un par un face à tous les autres masques en récupérant le Intersection Over Union (noté IoU par la suite). Le meilleur score est conservé car celui-ci signifie que c'est la cellule que l'on cherche à masquer.

Pour chaque masque, on a donc un IoU donnée. Pour tout un ensemble de seuil (allant de 0.5 à 0.95 par steps de 0.05), si l'IoU est supérieur à ce seuil, le masque est considéré comme un "hit". Pour chaque image, la moyenne des masques avec un "hit" est faite. C'est ce résultat qui est utilisé pour l'évaluation. Afin de limiter les ressources de calcul, seulement quelques masques sont testés.

# Exploration

## Type d'images

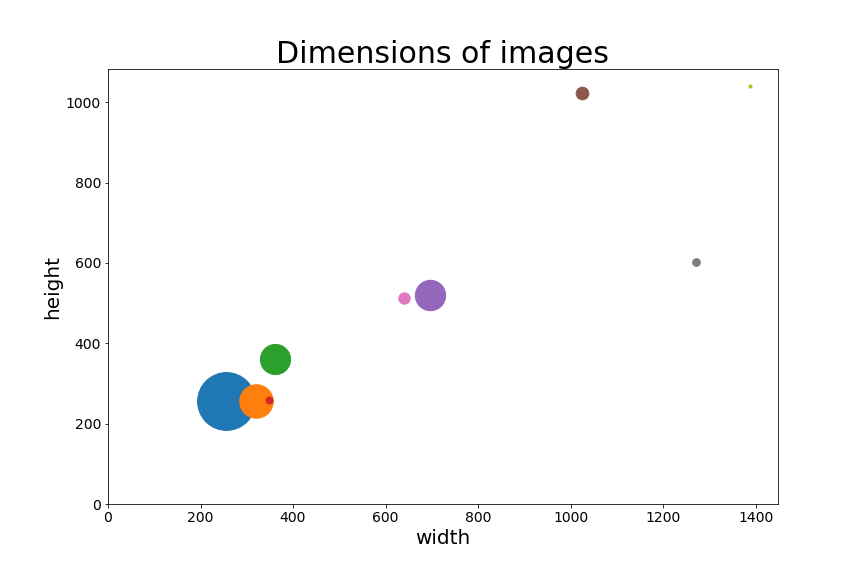
Lors de l'exploration, on peut remarquer que l'on a 3 types d'images. Certaines sont en couleurs, d'autres avec les cellules en blanc sur fond noir et d'autre avec la cellule en noir sur fond blanc.



Très globalement, on a majoritairement les images avec les cellules blanches sur fond noir.

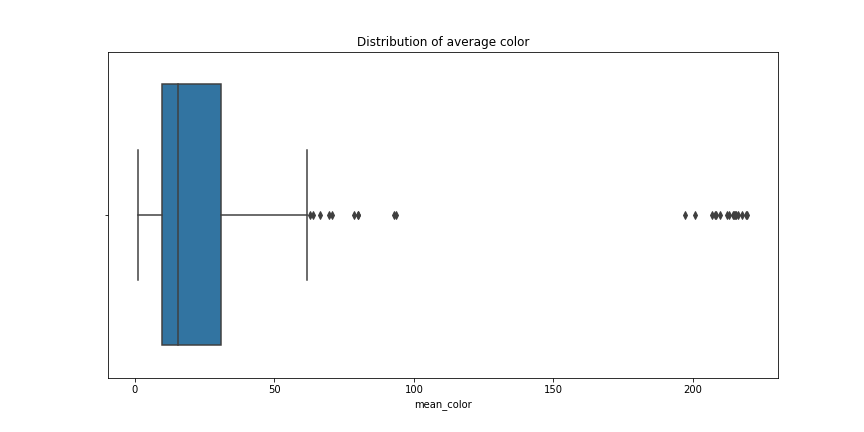
## Taille des images

Concernant la taille des images, elles sont assez variables mais globalement assez petites et avec des ratios correctes. On peut afficher les dimensions via un nuage de points comme ci-dessous:



## Couleur moyenne

Pour la phase de pré-processing, un des objectifs était de convertir ces 3 types d'images dans un seul type (fond noir et cellules en blanc). Pour ce faire, l'ensemble des images a été convertie en Noir et Blanc et la couleur moyenne a été récupérée. On peut voir le boxplot de celle-ci ci-dessous.



On peut remarquer que certaines images ont une moyenne très haute donc un majoritairement du blanc. A l'inverse, on a beaucoup d'images avec le noir majoritaire. On peut remarquer un gap important entre ces 2 types d'image et on peut donc faire l'inversion en fonction d'un seuil mis à 150 dans notre cas. On peut maintenant passer au pré-processing.

# Pré-processing

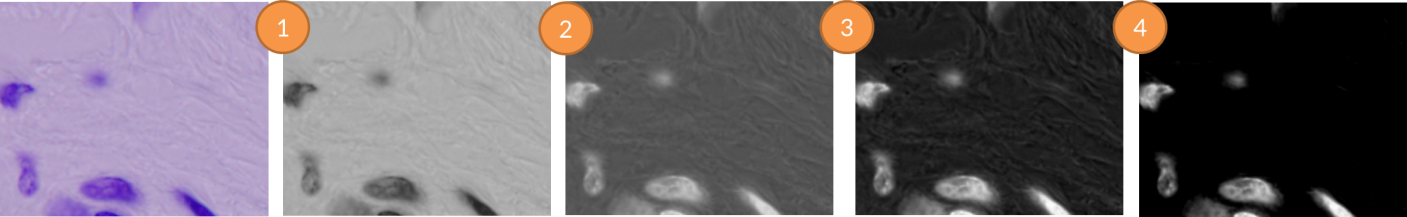
## Préparation des masques

Concernant les masques fournis, on a un dossier par images avec tous les masques de celle-ci. Pour la phase de pré-processing, une simple fusion a été faite afin d'avoir un masque global.



## Augmentation de Contraste

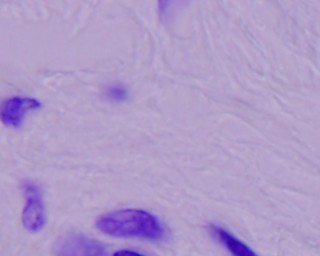
Comme on l'a vu précédemment, on a un seuil permettant de détecter si l'image est blanche sur fond noir ou noir sur fond blanc. Ajouté à cela, on va normaliser le constraste. L'objectif étant d'avoir le max à de l'image 255 et le minimum à 0. Cela a pour effet d'augmenter les contrastes. Par la suite, une seconde augmentation sera faite avec pour objectif dépasser les 255 et les 0 afin de clipper l'image. Sur une image test, cela donne :



La phase 1 est la conversion en Noir et Blanc, la phase 2 est l'inversion en Blanc et Noir (moyenne > 150). La phase 3 est le boost de contraste et la phase 5 est le seuillage. Cela permet d'avoir des images homogènes et mieux faire ressortir les noyaux.

## Adaptative Threshold

Lors de différents préprocessing testés (disponible dans le notebook Data Préparation), un type de pré-processing m'a aussi intéressé, c'est l'adaptative Threshold. Celui-ci permet de faire ressortir les contours de cellules et permettent ainsi d'aider à séparer les noyaux qui se touchent par la suite.





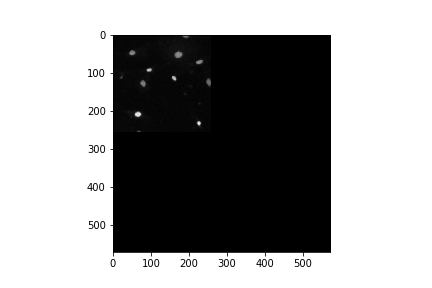
Ce dessus, se trouve à gauche l'image de base, au milieu l'image pré-processée et à droite le résultat de l'adaptative Threshold. Ce résultat étant intéressé pour un dernier modèle.

## Redimensionnement

Pour les CNN que l'on verra par la suite, on a besoin d'une image d'une dimension fixe. Selon les modèles, des images de 512 x 512 étaient trop lourdes en mémoire donc les images ont été redimensionné en 256 x 256. Pour ce redimensionnement, 2 approches ont été pensées.

### Version non déformée

Afin de ne pas impacter la forme des cellules, le redimensionnement imaginé au départ était de ne redimensionner que les images plus grandes que la taille voulue. Sinon elle sera positionnée dans un coin. Quant au redimensionnement, le facteur est constant pour avoir le plus grand coté égal à la taille souhaitée.



Ci-dessus, on a le cas d'une image de 256 x 256 pour une image voulue de 512x512. Malheureusement le fond n'étant pas toujours parfaitement noir, cela a créé beaucoup de bruits lors de l'entrainement et performait moins bien.

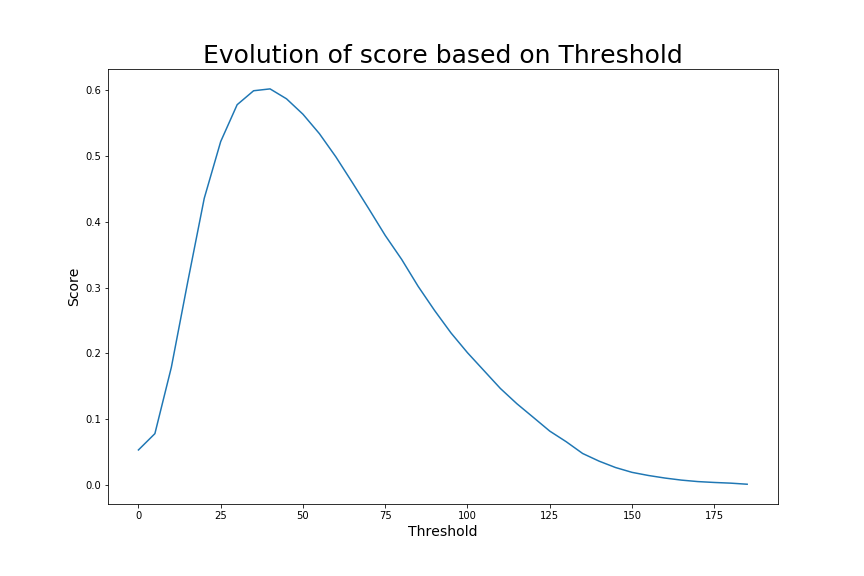
### Version classique

Suite à l'échec du précédent redimensionnement, et au vu des ratios (hauteur/largeur) qui ne sont pas extrêmes, un scaling classique a été utilisé. Dans un premier temps, afin de tester le modèle en 128 x 128 puis en 256 x 256.

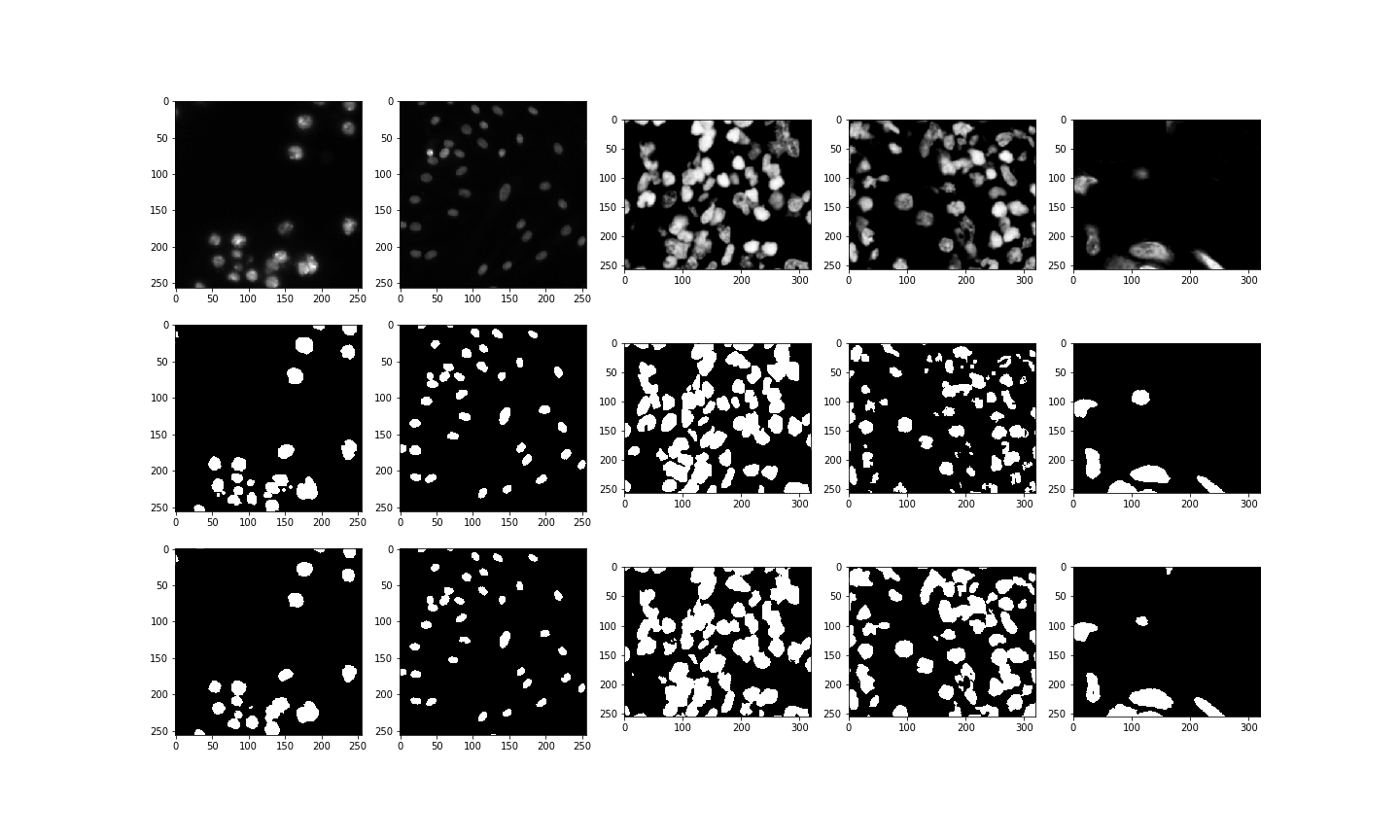
# Modèles

## Modèle Simple

Le premier modèle que l'on peut tester afin d'avoir une baseline, est un modèle classique n'utilisant que de l'image processing. Pour ce faire, nous avons déjà les images pré-processée prédemment. Sur celle-ci, un simple threshold sera appliquée sur la valeur du pixel sera faite. Si on prend un seuil trop bas, l'image aura trop de blanc et l'union sera très grande résultat à un faible IoU. Si on prend un seuil trop haut, on aura trop peu d'intersection et donc un score faible. De ce faite, une mesure du IoU **sur le masque complet** a été faite suivant tous les seuil possible. On trouve les résultats suivants :



Le seuil optimal est à 40. Basé la dessus, on peut prédire les masques et les comparer à ceux fournis. La 1ère ligne montre les images pré-processées, la 2nde ligne est le vrai masque et la dernière ligne est la prédiction.



On peut remarquer que les résultats sont assez similaires.

## U-net Classique

En 2015, un modèle de segmentation d'image dans le milieu médical a gagné le ISBI 2015 loin devant les autres modèles. Ce projet étant similaire, c'est le modèle le plus utilisé par les compétiteurs. C'est donc aussi un modèle que j'ai voulu testé.

Un des bémols sur ce projet avec le U-net classique (architecture présentée en Annexe 1), c'est qu'il coupe les bordures. Sur une image de 572x572, le masque est de 388x388. Afin d'avoir une taille identique en entrée qu'en sortie, j'ai décidé de prendre une taille multiple de 2 afin de ne pas avec de soucis lors du Max Pooling ainsi que de passer tous les padding en "same" (L'architecture est fournie en Annexe 2).

Ce modèle a été entrainé avec les images pré-processées pour avoir un nombre de layer fixe (à 1) redimensionné en 256 x 256. La data-augmentation a été testée mais ne donne pas de bon resultats pour les mêmes raisons que le redimensionnement sans déformations (bordures blanches). Quelques images augmentées sont présentes en annexe 4.

Un callback a été mis en place aussi pour enregistrer à chaque Epoch la prédiction sur une image de test. En fin de training un gif est généré avec l'ensemble des prédictions. Les gifs sont présent dans le dossier training sur github (<https://github.com/Coni63/OC_DS/tree/master/P9/img/training>). Le numéro correspond à l'ID de l'image est est contant. L'idée est de tester la prédiction sur des masques très différents.

Si on regarde ses gif, on peut remarquer que la prédiction n'est pas stable du tout avec souvent des masque qui deviennent tout blanc. Cependant, le modèle sauvegardé (après 50 Epoch) est **par chance** sans ce défaut. Selon les entrainements, on a parfois des cellules très fusionnées. Pour aider à les décomposer, un modèle extended a été testé.

## U-net Extended (ou multi-arm)

Sur ce modèle, l'objectif était d'utiliser l'image pré-processée mais aussi l'image de base et l'adaptative threshold vu au départ. Pour ce faire, la partie "convolution/encodage" a été dupliquée pour prendre ces 3 images en inputs. Et pour le transfert des tenseurs lors de la dé-convolution, c'est pareil sauf que la matrice de concaténation est plus grande. Le modèle est aussi représenté en Annexe 3.

Concernant l'entrainement, la data augmentation a aussi été supprimée pour les même raisons et le callback a aussi été mis en place (mais un peu plus complexe). Les gifs des entrainements sont aussi présent sur github à la même adresse. Dans les dossiers, les images des entrainements, ainsi que l'image de base et le masque sont disponibles.

# Post-processing

Comparé aux précédents projets, une phase de post-processing est nécessaire. En effet, nous prédisons un masque global pour l'image mais la compétition demande un masque par cellule.

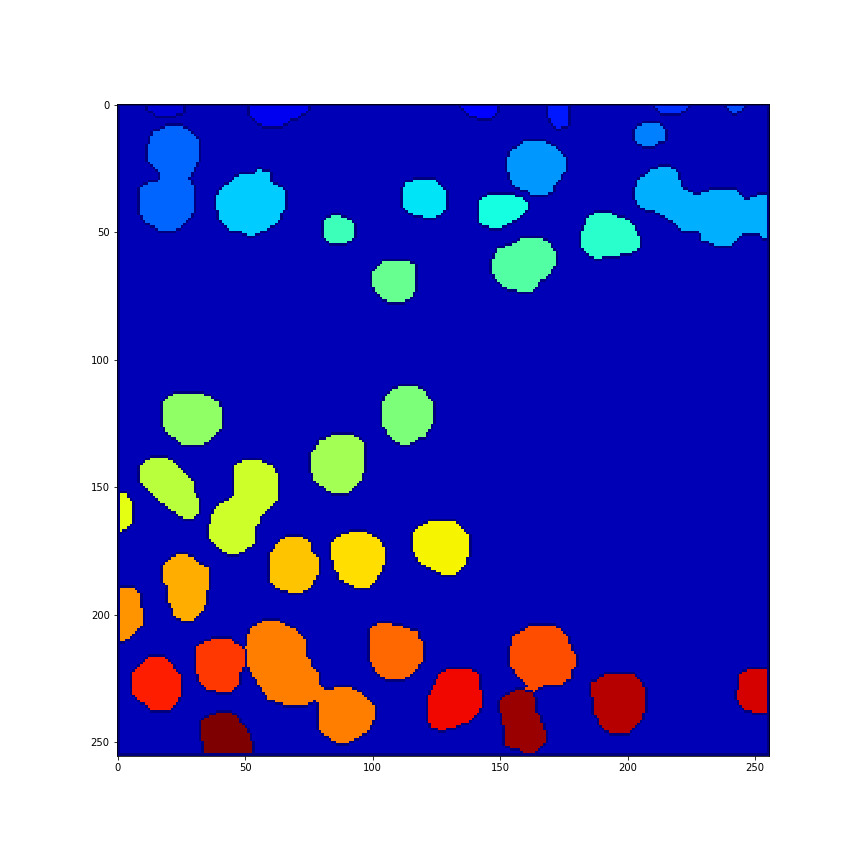
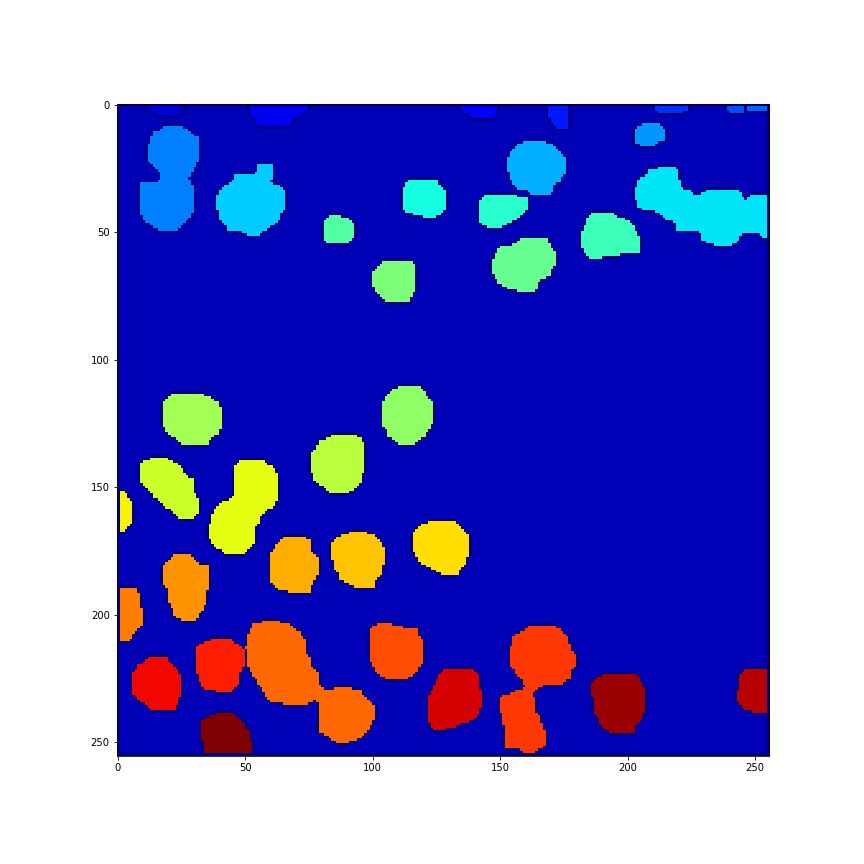
Dans un 1er temps, pour les CNN, il a fallu aussi déterminer le seuil à partir de laquelle on considère un pixel blanc ou noir (comme on l'a fait pour le modèle simple). En effet, on a en sortie une image basé sur une sigmoïde pour chaque pixel. Comme on l'a fit pour le modèle simple, l'étude du IoU en fonction du seuil a été fait sur le masque complet.

Ceci fait, on peut décomposer l'image par zone blanche qui se touche. Cela peut se faire avec diverses librairies comme skimage (équivalent de sklearn mais pour les images ou encore OpenCv). Cependant, cela n'est pas optimal car si on a 2 cellules très proche, il se peut qu'un ou 2 pixels les connectent et donc réduisent fortement le score (car il manque un des masques et l'union est plus grande). Pour améliorer la décomposition, on va appliquer différentes méthodes de la librairie OpenCv.

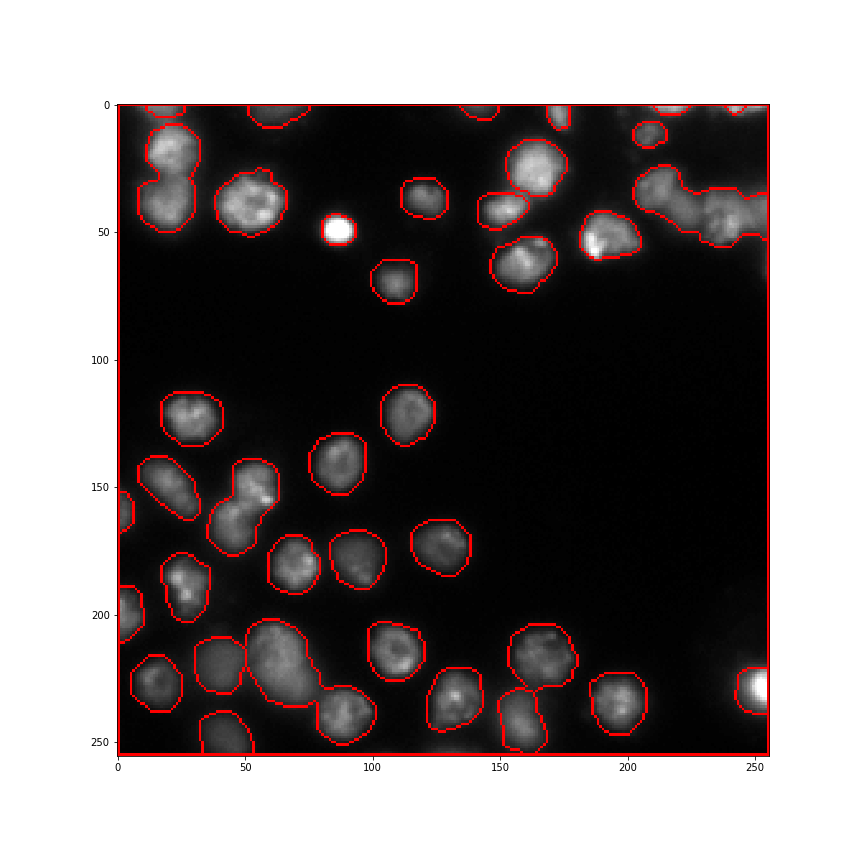
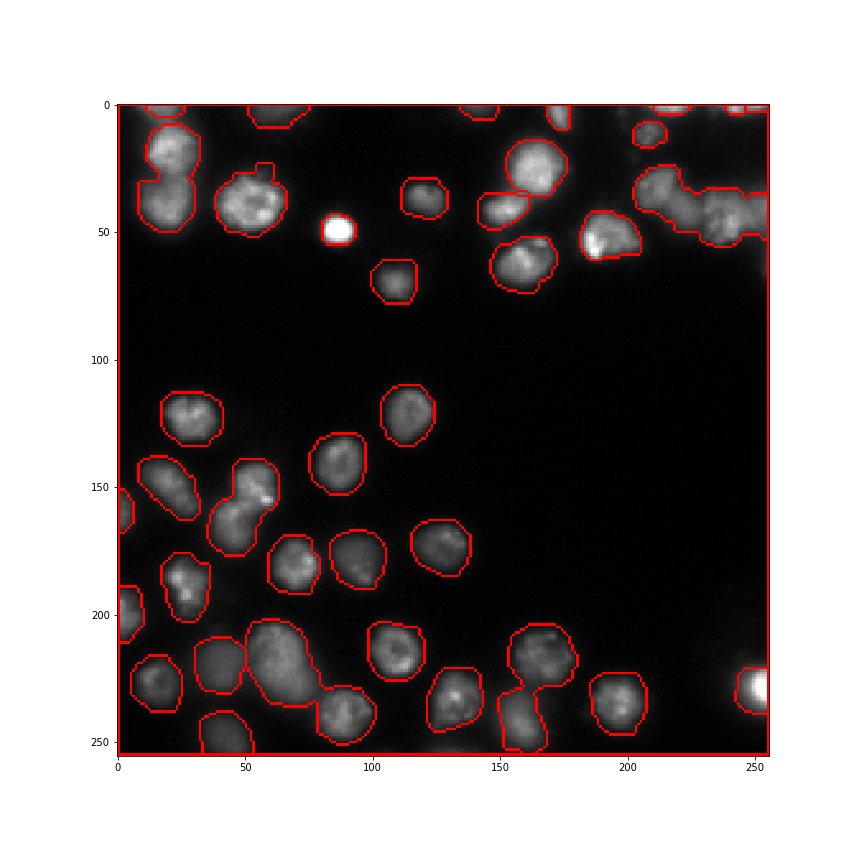
La 1ère étape est de supprimer le bruit (des pixels isolés qui seraient considérer comme des masques) Cela peut se faire avec un érode puis dilate. Si la cellule est plus grande que Erode, on la garde à l'identique, sinon elle disparait.

On peut ensuite appliquer un filtre oval pour lisser les prédictions sur des formes ovales. Cela n'est pas forcément nécessaire mais aide bien à la décomposition (ce gain sera montré à la fin)

Ensuite, il faut calculer les distances inter pixels pour trouver les centres des cellules. Par rapport aux distances, on peut décomposer 2 cellules qui se touchent avec la méthode 'connected Components' de OpenCv. On a donc maintenant 1 masque par cellule (à gauche sans le filtre oval et à droite avec celui-ci). On peut remarquer qu'avec, la cellule en bas à gauche se trouve aussi décomposée.



Ou encore en superposition de l'image de base:



Sur github, vous trouverez aussi les différentes prédictions faite sur l'image de base avec les contours des cellules avec les différentes méthodes dans les dossiers extractions présents à [cette adresse.](https://github.com/Coni63/OC_DS/tree/master/P9/img)

# Analyse des résultats

# Pistes d'évolutions

# Conclusion

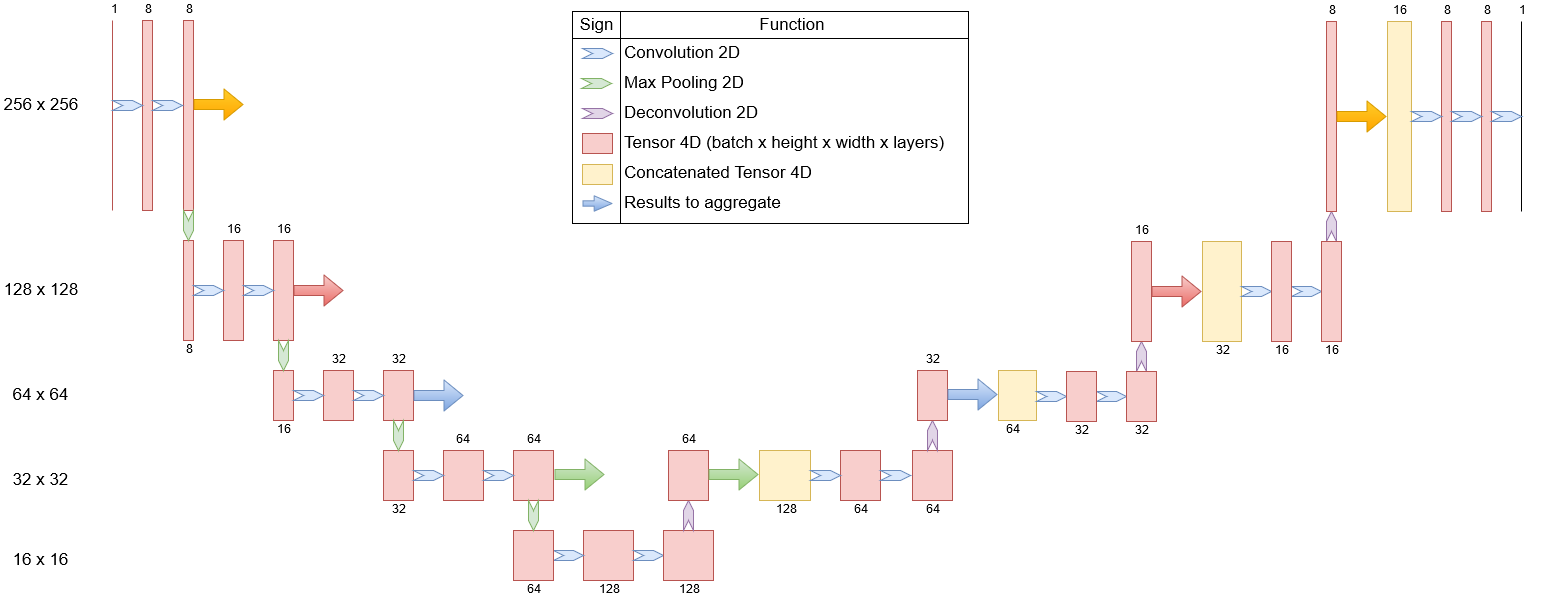
# Sources

# Annexe

### Annexe 1 : Structure du U-net Standard

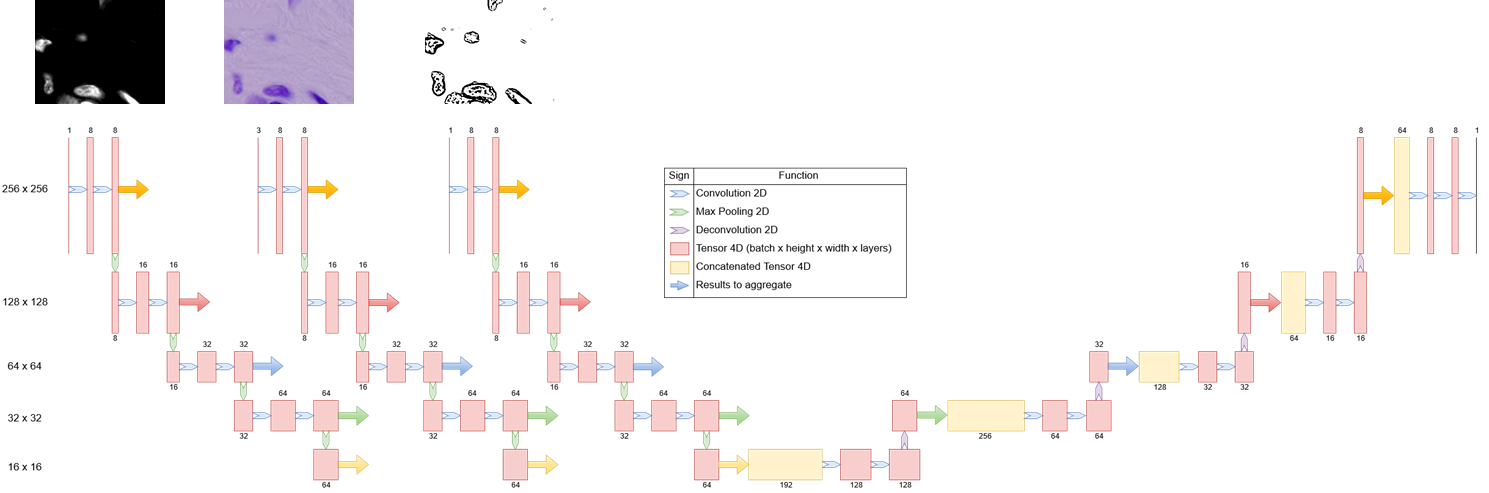


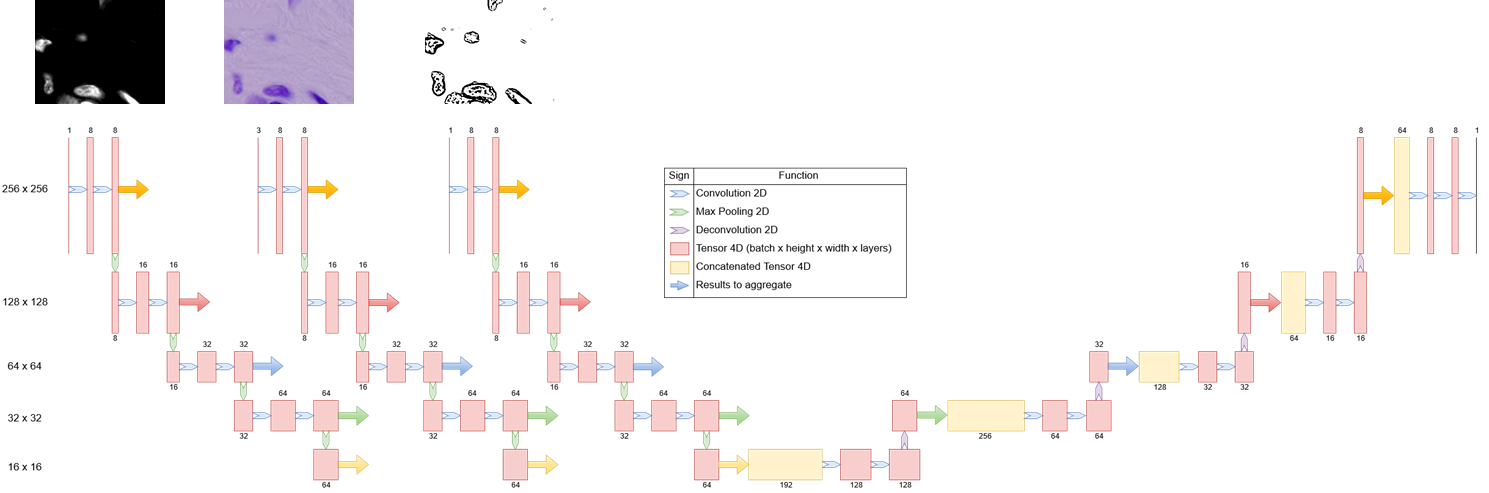
Annexe 2 : Structure du U-net modifié



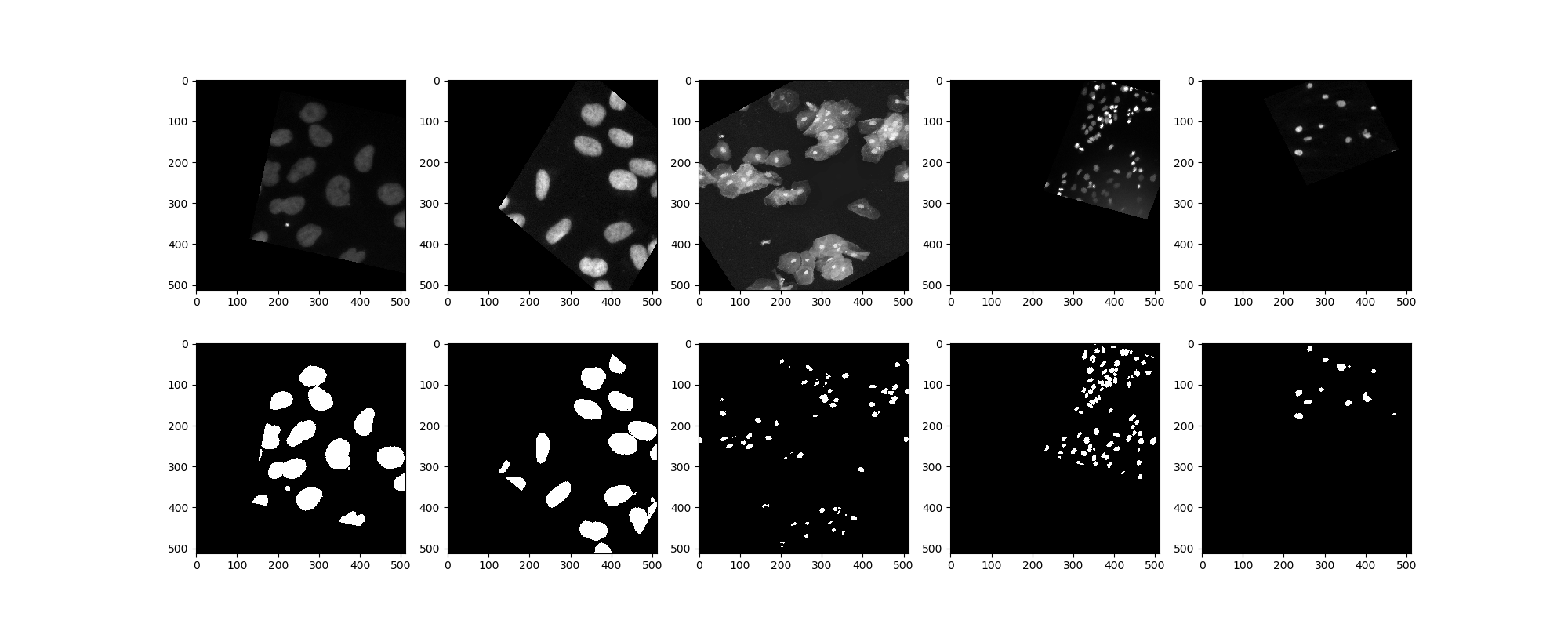
Annexe 3 : Structure du U-net extended

Pour des raisons de lisibilité, le modèle a été coupée. Chaque bras de convolution sont représentés en haut et la partie déconvolution est en dessous.

****

****

Annexe 4 : Data Augmentation



On peut remarquer avec cette data augmentation les bords d'images qui apportent trop de bruits lors du training.