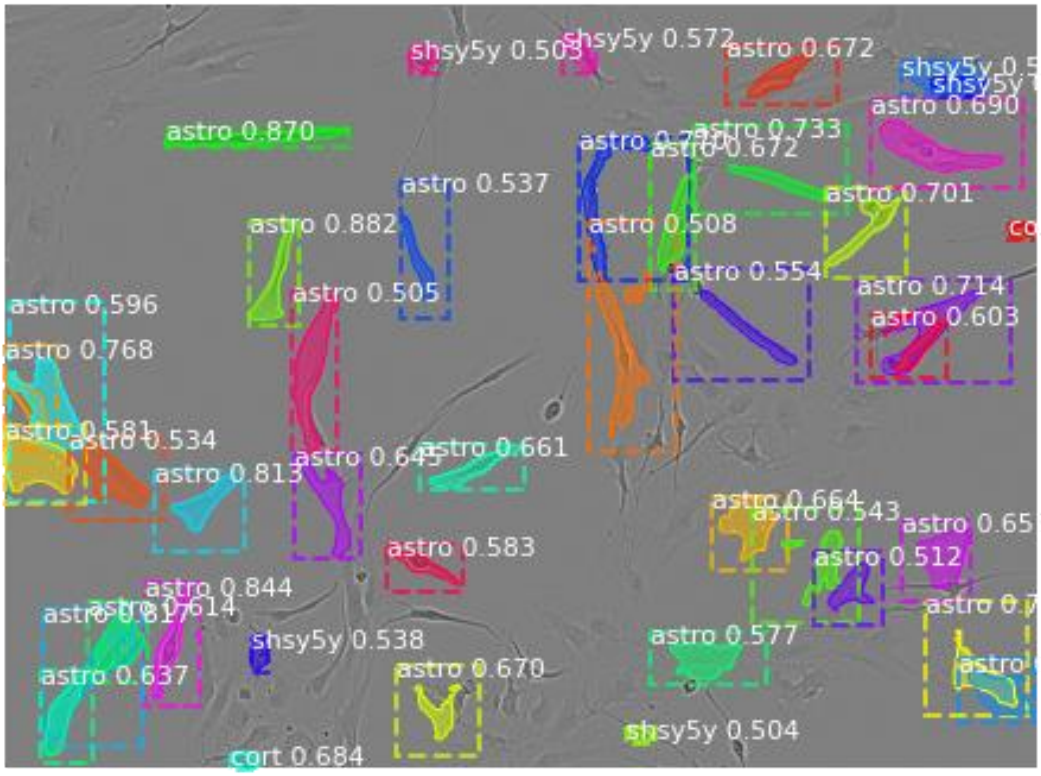
P8
Compétition Kaggle
‘Instance segmentation’ sur des images de cellules
-
Rapport



Introduction

Le but de ce projet est de participer à une compétition Kaggle et de présenter les travaux associés. Le projet choisi est directement dans la lignée des projets P6 et P7, qui traitaient respectivement de la classification d'images et d'une veille technologique sur la détection d'objets.

Plus précisément, la compétition choisie met en œuvre de la 'segmentation d'instances' appliquées à des images de cellules neuronales. Elle permet de réutiliser les travaux du P7 tout en allant plus loin.

Sommaire

I. Présentation de la compétition Kaggle choisie

Lien vers la compétition : Sartorius - Cell Instance Segmentation | Kaggle

Sujet : Cette compétition est organisée par Sartorius, qui est une entreprise qui travaille en collaboration avec la recherche médicale et qui emploie des ingénieurs et data scientists pour appuyer les chercheurs dans leurs travaux.

La compétition met à disposition des images de cellules neuronales par contraste de phases. **Le but de la compétition est d'identifier et de localiser chaque cellule au sein de l'image.** Il y a 3 types de cellules distincts :

- Les neurones, classe correspondante : '*cort*'
- Les cellules de type 'sh-sy5y', classe correspondante '*sh-sy5y*'
- Les astrocytes, de classe correspondante '*astro*'

Point important : Chaque image contient uniquement un type de cellules.

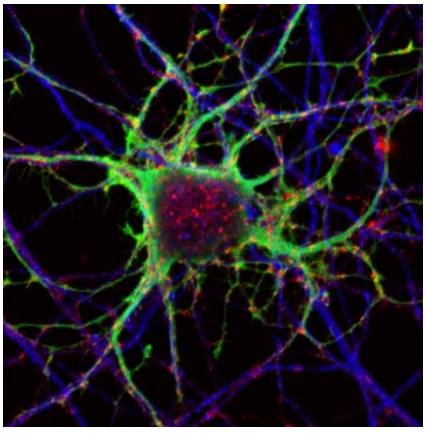
Lien vers le code implémenté (git) : https://github.com/Rickyviii/M-RCNN-for-Kaggle-Cells-competition/blob/4b8ba5c6fd95e59fa03c9502418b1d65af2eceaa/kaggle_competition_cells.ipynb

Lien vers le code implémenté (Kaggle) : Kaggle Competition _ with TensorFlow Mask-RCNN

I.1 Les 3 types (ou classes) de cellules

Les cellules neurones (classe 'cort') : Extrait de The Neuron (brainfacts.org)

> *"Cells within the nervous system, called neurons, communicate with each other in unique ways. The neuron is the basic working unit of the brain, a specialized cell designed to transmit information to other nerve cells, muscle, or gland cells."*



'Neurons, confocal fluorescence microscopy.' Source : ici.

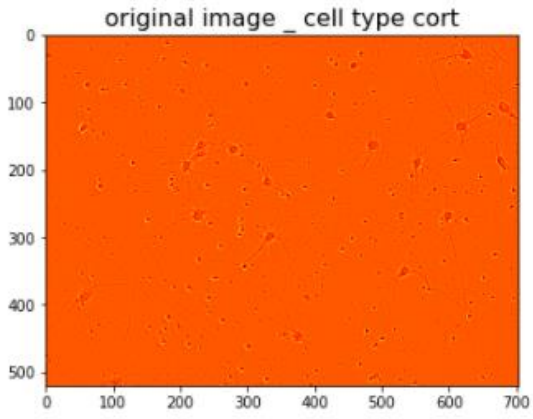
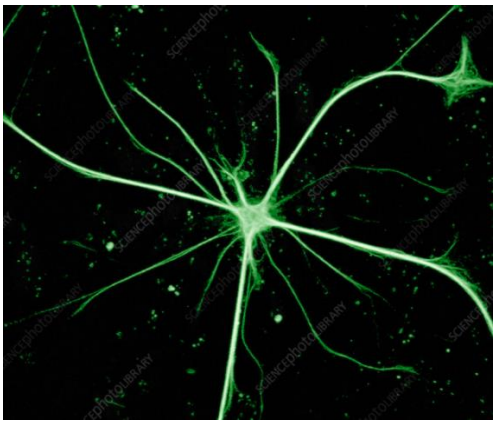


Image de classe 'cort' (neurones) telle que fournie par la compétition.

Les astrocytes (classe 'astro') : Extrait de What are Astrocytes? (news-medical.net)

> *"What are astrocytes?*
>
> *Astrocytes are highly heterogeneous neuroglial cells with distinct functional and morphological characteristics in different parts of the brain. They are responsible for maintaining a number of complex processes needed for a healthy central nervous system (CNS)."*



'Astrocyte nerve cell. Confocal light micrograph of an astrocyte nerve cell.' Source : ici.

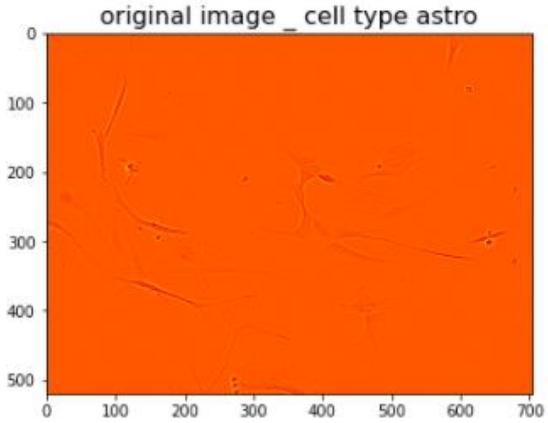
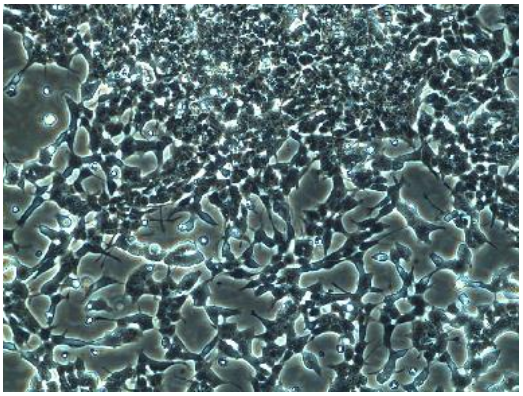


Image de classe 'astro' (astrocytes) telle que fournie par la compétition.

Les cellules sh-sy5y : Extrait de SH-SY5Y - Wikipedia

> *"SH-SY5Y is a human derived cell line used in scientific research. The original cell line, called SK-N-SH, from which it was subcloned was isolated from a bone marrow biopsy taken from a four-year-old female with neuroblastoma."*

En d'autres termes, ce sont des cellules artificielles provenant d'une série de clonages de cellules nerveuses d'une personne atteinte de neuroblastome, un type de tumeur du cerveau.



The SH-SY5Y cells are dense and considered to be confluent at this stage. Source : ici

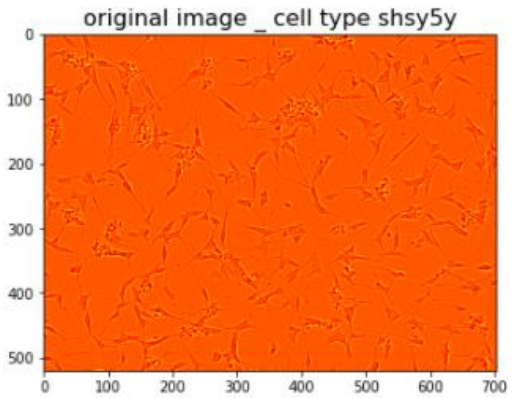


Image de classe 'sh-sy5y' telle que fournie par la compétition.

I.2 Domaines techniques et algorithme retenu

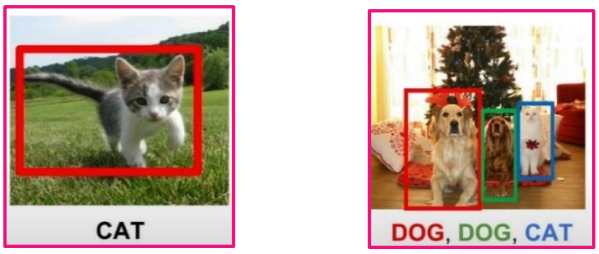
I.2.1. Domaines techniques de la computer vision

On distingue plusieurs approches dans la 'computer vision' :

- **Image classification :**
  - On a une image en entrée dont il faut prédire en sortie la classe d'appartenance

Exemple de classifieur de race de chiens.

- **Simple object detection :**
  - On identifie les objets appartenant à une des classes de départ par un rectangle (localisation + taille), que l'on appellera bbox, associés à la classe prédite.
  - Le nombre d'objets distincts à prédire est à priori inconnu et dépend de l'image :



Object detection, un 'objet' / plusieurs 'objets' (Source ici)

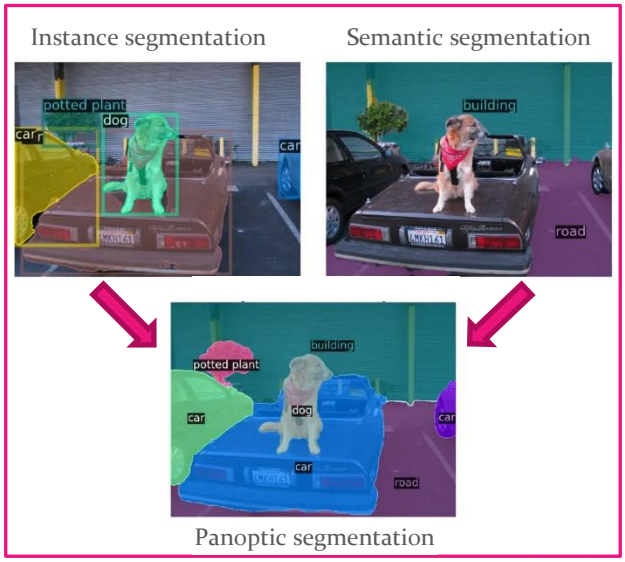
- **Semantic segmentation :**
  - Ici, on ne parle pas d'identification d'objets au sens propre, mais d'identification de pixels associés chacun à une classe spécifique. Il n'y a pas de notion de localisation d'objets, juste une classification d'appartenance de chaque pixel.

Semantic segmentation (source ici)

- **Instance segmentation :**
    - Ici, on associe des données de localisation et de classification d'objets avec une vue pixel par pixel. On aura donc en sortie, par image, une liste d'objets, de bbox associées (taille + position), et un masque représentant en noir et blanc l'objet en question : chaque pixel de la bbox est donc associé à la catégorie identifiée ou pas.
    - On peut en déduire aisément, comme dans le cas précédent, le nombre de pixels de l'objet identifié.

Instance segmentation (source ici)

- **Panoptic segmentation :**
    - C'est une fusion de l'instance segmentation et de la semantic segmentation.
    - Chaque pixel doit faire l'objet d'une attribution à une classe



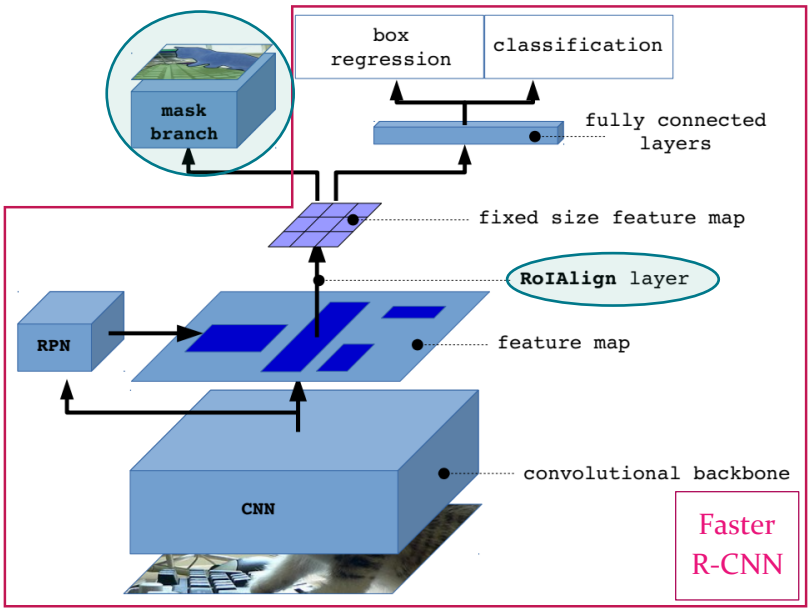
Panoptic segmentation (source ici)

I.2.2. Domaine technique et algorithme retenu

Ici, il est expressément demandé d'identifier par localisation de pixels chaque cellule d'une image. On rentre donc dans le cadre de la segmentation d'instances, où l'on devra identifier toutes les instances représentant des cellules, et les localiser par des masques de pixels.

Algorithme retenu : M-RCNN

- Papier de recherche officiel : [1703.06870] Mask R-CNN (arxiv.org)
- Auteurs : Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick



Mask R-CNN (source ici)

Le Mask R-CNN est issu d'un autre algorithme appelé Faster R-CNN, auquel on rajoute **une brique supplémentaire de prédiction de masque**.

Il permet de faire de la 'instance segmentation' en fournissant en sortie, en plus des outputs du Faster R-CNN (classification d'instances + détection d'objets), **le masque de chaque objet détecté**.

Il est donc clairement indiqué pour la problématique de cette compétition : identifier des instances (donc des masques) de cellule spécifiques dans des images, via 3 classes distinctes proposées.

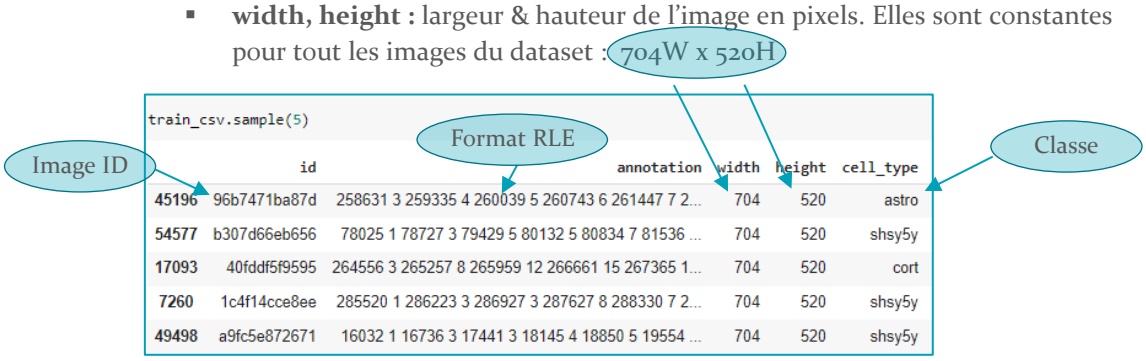
II. Visualisation des Données en Input et Output

II.1 Input : données d'entraînement

II.1.1. Présentation

Les données fournies en input sont composées de :

- **606 images** de cellules au format png, dont le nom est formatté comme suit : '*<id_img>*.png'
- Un dataset '*train.csv*' composé de 73585 entrées et 9 colonnes :
  - Chaque ligne représente une instance (donc une cellule).
    - **On a donc en tout 73585 cellules réparties sur 606 images**
  - 5 champs sont utiles parmi les 9 :
    - **id :** l'id de l'image de rattachement de l'instance, cad l'id de l'image ou est présente la cellule. On a donc plusieurs entrées pour une seule image
    - **annotation :** le codage du masque au format RLE (Real Length Encoding), qui représente la localisation exacte des pixels composant la cellule dans l'image
    - **cell_type :** la classe de la cellule, càd son type ('astro', 'shsy5y' ou 'cort')
    - **width, height :** largeur & hauteur de l'image en pixels. Elles sont constantes pour tout les images du dataset : 704W x 520H



`train_csv.sample(5)`

	id	annotation	width	height	cell_type
45196	96b7471ba87d	258631 3 259335 4 260039 5 260743 6 261447 7 2...	704	520	astro
54577	b307d66eb656	78025 1 78727 3 79429 5 80132 5 80834 7 81536 ...	704	520	shsy5y
17093	40fddf5f9595	264556 3 265257 8 265959 12 266661 15 267365 1...	704	520	cort
7260	1c4f14cce8ee	285520 1 286223 3 286927 3 287627 8 288330 7 2...	704	520	shsy5y
49498	a9fc5e872671	16032 1 16736 3 17441 3 18145 4 18850 5 19554 ...	704	520	shsy5y

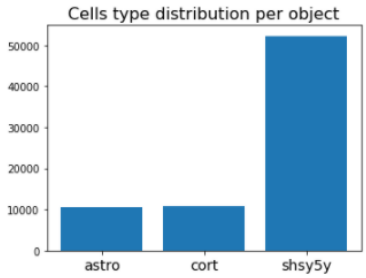
Format des données en Input

II.1.2. Analyse et visualisation

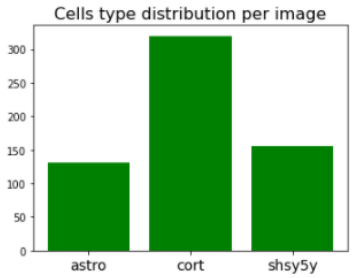
Distribution

Les graphes suivants montrent :

- La distribution des instances par classe (toutes images confondues)
- La distribution des images par classe
- Le nombre d'instances, selon leur classe, par image (i.e. la densité de cellules selon la classe)



Nombre d'instances par classe



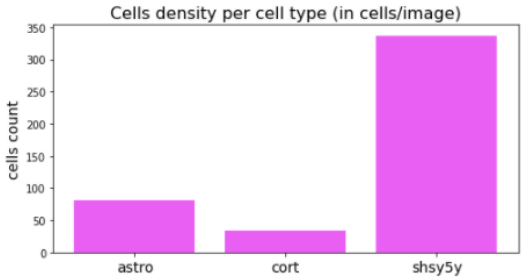
Nombre d'images par classe

Beaucoup d'images 'cort' pour peu d'instances. Au contraire, beaucoup d'instances 'shsy5y' pour un nombre d'images moyen.

- ⇨ Les images contenant des neurones 'cort' sont très peu denses en cellules.
- ⇨ Les images contenant des cellules clonées 'shsy5y' sont très denses en cellules.

Effectivement, en moyenne :

- 121 cellules / image
- Cellule astro : 80,3 cellules / image
- Cellule cort : 33,7 cellules /image
- Cellule shsy5y : 337 cellules/image

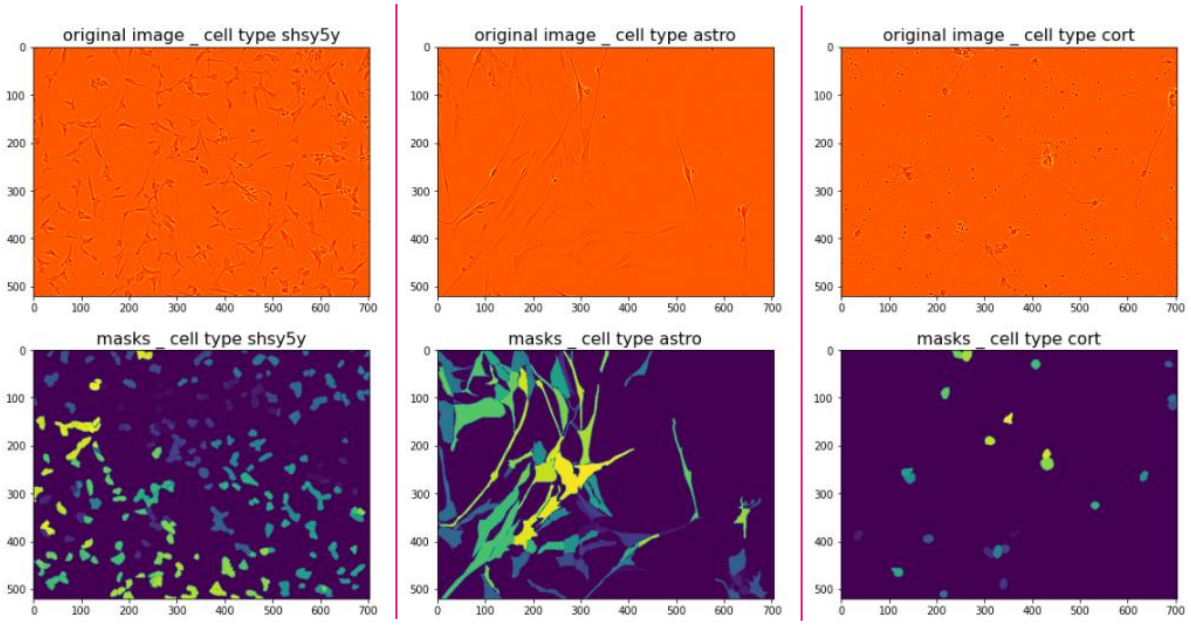


Densité de cellules par classe dans une image

Encodage des images et masques, et visualisation

Les images sont encodées au format png, mais au format 'grayscale' (L), pas en RGB.
Les masques sont présents sous la colonne annotation, au format RLE.
On utilise plusieurs fonctions issues d'un code Kaggle (source ici, §3) pour décoder les masques et les représenter sous forme de tableaux multidimensionnels, exploitables pour la représentation graphique. Ces fonctions, dans mon code, ont été adaptées.

Voici ici la représentation graphique de 3 images (une par classe), avec les masques attendus en sortie. Les 3 images ont été prises au hasard dans le jeu d'entraînement.



Pour chaque classe : image d'entrée, et en dessous, visualisation des masques attendus en sortie

II.2 Output : données attendues et scoring

Format attendu pour la soumission :

La compétition Kaggle attend les données prédites, à savoir les masques, au format RLE, avec une entrée par instance, et l'id de l'image de rattachement.

C'est donc exactement le même format que les données fournies en entrée pour l'entraînement du modèle via les 2 premières colonnes du fichier '*train.csv*'. Le fichier de soumission sera donc un fichier csv avec 2 colonnes 'id' et 'predicted' :

```
Id, predicted
0030fd0e6378, 118145 6 118849 7 119553 8 120257 8 120961 9 121665 10 122369 12 123074 13 123778 14 …
0030fd0e6378, 17652 2 18355 5 19059 6 19763 7 20467 8 21170 10 21874 11 22578 12 23282 14 23986 15 …
0030fd0e6378, 178698 6 179402 7 180106 8 180810 8 181514 9 182218 10 182922 11 183626 12 184330 12 …
…
0140b3c8f445, 300269 2 300971 5 301672 8 302374 10 303075 13 303777 14 304479 16 305182 16 305884 17 …
0140b3c8f445, 208385 1 209089 2 209793 3 210497 3 211201 4 211905 5 212609 7 213313 10 214018 11 214722 14 …
```

Format attendu du fichier csv de soumission

NB : il n'est pas demandé ici de prédire les classes des instances (leur type), juste leur emplacement !

Prérequis supplémentaire de la compétition : tous les masques prédits doivent être sans recouvrement entre eux (pas d'overlap), alors que ceux fournis pour l'entraînement (masques vrais) se recouvrent partiellement.

Evaluation/scoring :

Le score total du fichier fourni est calculé comme la moyenne des scores de chaque image.

$$Score = Mean_{(t,img)}\left(AP_{t,img}\right)$$

Pour une image donnée, chaque score est lui-même calculé comme une 'Average Precision' (AP) moyenne. La notion d'AP fait en fait appel à celle de matrice de confusion :

$$AP_{t,img} = \frac{TP_{t,img}}{TP_{t,img} + FP_{t,img} + FN_{t,img}}$$

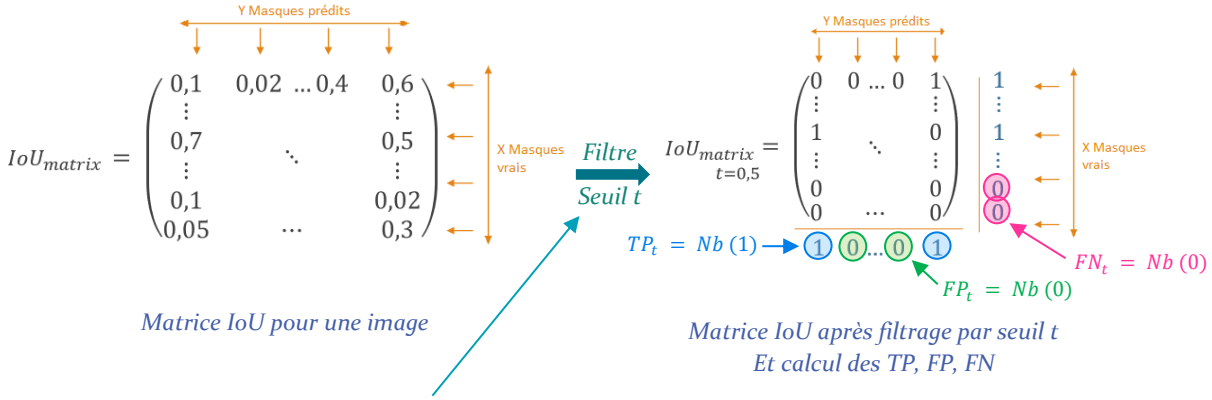
Où TP = True positive (vrai positif), FP = Faux positif, et FN faux négatif.
TP + FP + FN = l'ensemble des détections effectuées.

Dans le cas d'une segmentation d'instances, il va s'agir de définir ce qu'est une vraie prédiction (TP), un faux positif (FP) et un faux négatif (FN) pour une image composée de plusieurs instances.

On a une prédiction quand un masque attendu et un masque prédit se recouvrent 'suffisamment' :

- La notion 'suffisamment' fait intervenir celle de 'seuil' t.
- Le recouvrement est lui mesuré par une métrique appelée IoU (Intersection Over Union, ou taux de recouvrement)
- Prédiction = IoU >t

⇨ Comme on a plusieurs masques vrais/prédits, on va créer une matrice IoU, avec X lignes représentant des masques vrais, et Y colonnes représentant les masques prédits.



On va ensuite appliquer un filtre seuil t (*[IoU>t] => 1, [IoU<=t] => 0*) à cette matrice.

Le nombre de 1 et de 0 va nous permettre de déduire les valeurs TP, FN, FN pour le seuil t, et donc la valeur d'AP pour l'image donnée et le seuil t.

On fait ensuite varier t sur le range (0.5, 0.55, ..., 0.95), et on prend la moyenne de l'AP sur l'ensemble des seuils t.

En résumé, le score est une moyenne sur toutes les images et toutes les valeurs de seuil, de l'Average Precision des instances prédites vs les instances vraies de chaque image.

III. Implémentation et premiers résultats

L'implémentation est effectuée via Mask-RCNN.

Le code utilisé pour mettre en place le M-RCNN est celui de Leekunhee (git source ici), qui est une version compatible Tensorflow 2.3 du code de Matterport (git source ici). Le modèle préentraîné est également issu de Matterport (fichier source ici) au format .h5

C'est ce même code qui a été déjà utilisé pour le projet 7, mais l'implémentation va ici plus loin : on supprime en effet les couches hautes du code pour entraîner le modèle sur nos 3 nouvelles classes et notre set d'entraînement composé des 606 images et instances associées.

III.1 Principales étapes :

- Création d'un dataset de training (80%) et de validation (20%) avec stratification par classes.
- Décodage des masques RLE (via fonction récupérée ici et adaptation) en tableaux multi-dimensionnels booléens (0/1), au format HEIGHT x WIDTH x NB_INSTANCES, représentant NB_INSTANCES sous format 1 (pixel noir) / 0 (pixel blanc)
- Chargement des données dans le code Matterport / Leekunhee
    - Création des classes
    - Création des masques attendus associés à chaque image

```
test_ = CellDataset()
test_.load_cells('train')
```

- Chargement du modèle M-RCNN pré-entraîné Matterport, sans les couches hautes

```
Cconfig = CellConfig()
my_model =  mrcnn.model.MaskRCNN(mode="training",
                                 config=Cconfig,
                                 model_dir = LOGS_DIR)  #class MaskRCNN for mrcnn/model.py

# Exclude the last layers because they require a matching number of classes
my_model.load_weights(COCO_MODEL_PATH, by_name=True, exclude=[
    "mrcnn_class_logits", "mrcnn_bbox_fc",
    "mrcnn_bbox", "mrcnn_mask"]) #exclude: list of layer names to exclude
```

- Rajout de nouvelles couches Fully Connected au modèle et entraînement sur nos données

```
Cconfig = CellConfig()
CDataset = CellDataset()

train_cell(my_model, Cconfig)
print((time.time() - st)/3600, 'h')
```

- On charge ensuite les poids du modèle entraîné (.h5) automatiquement stocké et nouvellement créé, en mode 'inference' (prédiction).

```
my_model_inf0 = mrcnn.model.MaskRCNN(mode="inference",
                                     config=CellconfigInf,
                                     model_dir = LOGS_DIR)

my_model_inf0.load_weights(MODEL_H5, by_name=True)
```

- On effectue nos prédictions sur de nouvelles images.

```
#PRED
img_id = df_img.loc[269, 'id']
img_name = img_id + '.png'
image = Image.open(os.path.join(DIR_IMG_TRAINING_RGB, img_name))

results = my_model_inf0.detect([np.array(image)], verbose=0)
NB_inst_pred, array_pred_multimasks, pred_cell_types = get_ouput_info(results)
plt.imshow(array_pred_multimasks, cmap = 'hot');
```

- On supprime l'overlap entre masques prédits sur chaque image (via fonction récupérée de ce code Kaggle)

```
_, array_pred_multimasks_woo, _ = get_ouput_info(results, remove_overlap=True)
```

- On calcule le score de chaque image prédite (l'AP moyen sur tous les seuils) ou d'un ensemble d'images prédites via fonctions récupérées et adaptées de ce code Kaggle.

```
#AP score for one image, when overlap is removed
iou_map([array_true_multimasks], [array_pred_multimasks_woo], verbose=1);
```

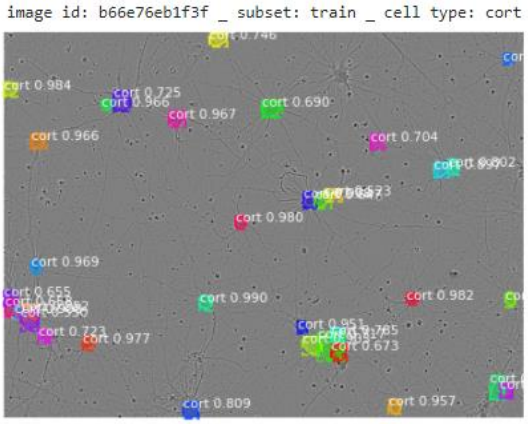
III.2 Premiers résultats

L'entraînement du modèle, sans data augmentation, au vu du nombre faible d'images mais important d'instances, a pris plusieurs heures. On observe ici des résultats sur des cellules du dataset d'entraînement ou de validation, dont on connaît les masques 'vrais'.

Hyperparamètres choisis dans cette configuration :

Learning rte = 1^{e-4}, Learning momentum = 0.9, Nb_epochs = 28

Prédictions sur des cellules de type 'cort' (neurones) :



Output des prédictions de l'algorithme Matterport

Thresh	TP	FP	FN	Prec.
0.500	20	15	16	0.392
0.550	19	16	17	0.365
0.600	17	18	19	0.315
0.650	17	18	19	0.315
0.700	16	19	20	0.291
0.750	13	22	23	0.224
0.800	7	28	29	0.109
0.850	3	32	33	0.044
0.900	0	35	36	0.000
0.950	0	35	36	0.000
Average Precision (AP)				
AP	-	-	-	0.206

Résultats

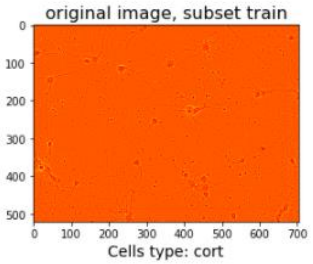
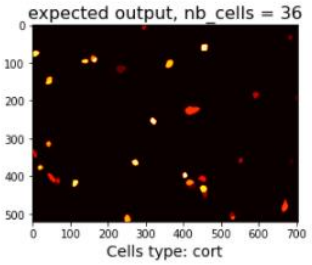
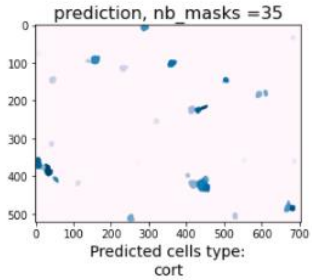


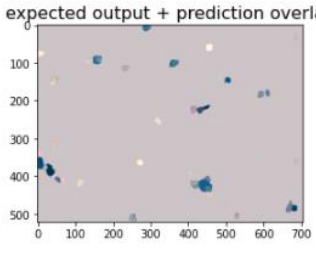
Image d'origine



Masques vrais

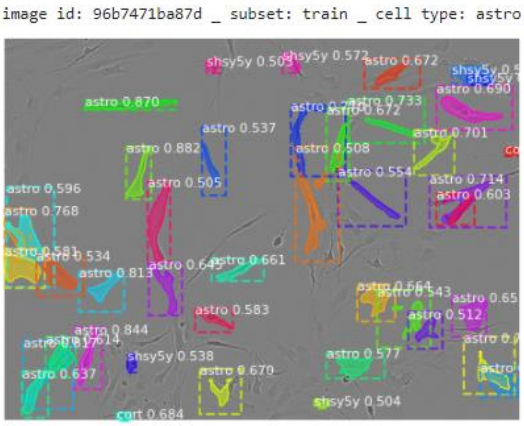


Masques prédits



Masques prédits + vrais

Prédictions sur des cellules de type ‘astro’ (astrocytes) :



Output des prédictions de l’algorithme Matterport

Thresh	TP	FP	FN	Prec.
0.500	10	32	123	0.061
0.550	8	34	125	0.048
0.600	8	34	125	0.048
0.650	7	35	126	0.042
0.700	6	36	127	0.036
0.750	3	39	130	0.017
0.800	0	42	133	0.000
0.850	0	42	133	0.000
0.900	0	42	133	0.000
0.950	0	42	133	0.000

Average Precision (AP)				
AP	-	-	-	0.025

Résultats

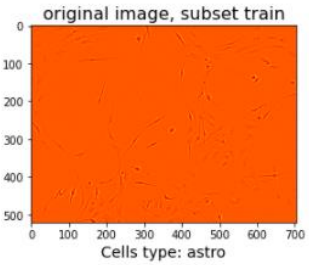
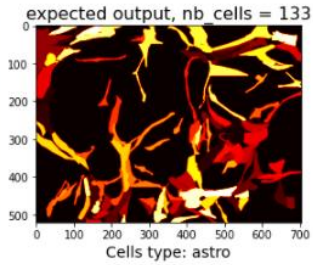
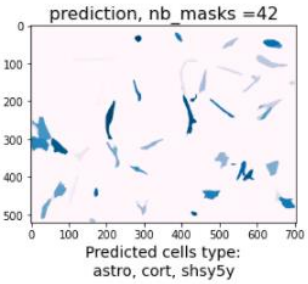


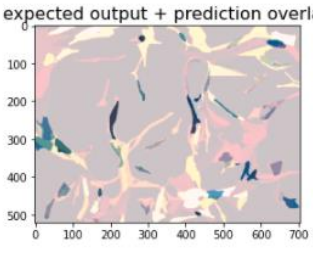
Image d’origine



Masques vrais

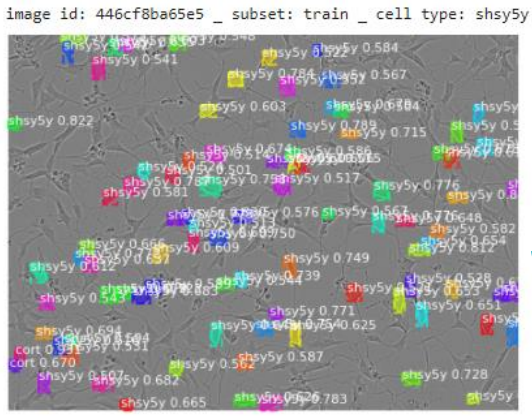


Masques prédits



Masques prédits + vrais

Prédictions sur des cellules de type ‘sh-sy5y’ (cellules clonées) :



Output des prédictions de l’algorithme Matterport

Thresh	TP	FP	FN	Prec.
0.500	62	28	325	0.149
0.550	56	34	331	0.133
0.600	47	43	340	0.109
0.650	44	46	343	0.102
0.700	33	57	354	0.074
0.750	19	71	368	0.041
0.800	10	80	377	0.021
0.850	3	87	384	0.006
0.900	1	89	386	0.002
0.950	0	90	387	0.000

Average Precision (AP)				
AP	-	-	-	0.064

Résultats

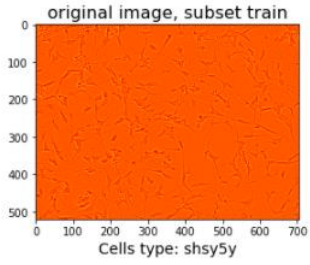
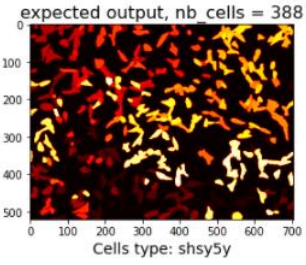
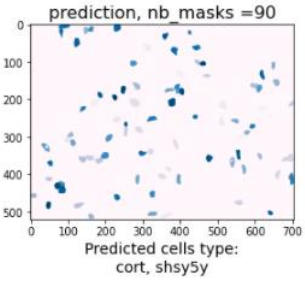


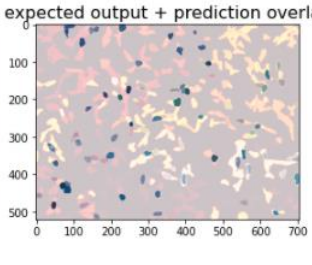
Image d’origine



Masques vrais



Masques prédits



Masques prédits + vrais

En résumé, les résultats ne sont pas bons sur ce premier test, notamment sur les images contenant des cellules de type 'astro' et 'shsy5y' : l'algorithme tel qu'il a été entraîné ne reconnaît pas suffisamment de cellules, ce qui conduit à un AP moyen assez faible.

Point important à considérer : les meilleurs résultats publiés sur le site de la compétition Kaggle montrent que les meilleurs APs globaux se situent aux alentours de 0,35. On est donc loin d'un résultat optimum de 0,8 ou plus.

III.3 Pour la suite

Ce qui est prévu pour améliorer les résultats :

- Augmenter la taille des données d'entraînement :
    - Via data Augmentation
    - En utilisant des images supplémentaires fournies pas la compétition dans un autre dossier
- Tester plusieurs valeurs d'hyperparamètres dont le learning rate
- Améliorer le procédé de transfer learning : dégeler plus de couches hautes pour améliorer l'entraînement des classes.

Conclusion

Cette compétition s'inscrit dans le domaine médical et correspond parfaitement à une problématique d'instance segmentation, ce qui a conduit au choix de l'algorithme M-RCNN pour répondre au besoin exprimé, à savoir l'identification de 3 types distincts de cellules nerveuses à partir d'images « à contraste de phase ».

L'identification doit se faire sans recouvrement des masques prédits (alors que les masques vrais servant à l'entraînement, eux, se recoupent).

C'est un sujet complexe avec une technique de scoring assez punitive : Le moindre écart entre les masques prédits et attendus fait chuter rapidement le score d'*Average Precision*, issu d'un calcul complexe à partir de 2 notions de base, le IoU (taux de recouvrement 'masque prédit / vrai') et les faux positifs, vrais positifs et faux négatifs estimés selon un seuil donné.

La complexité de cette compétition réside en 2 facteurs :

- Utiliser un algorithme de M-RCNN qu'il faudra entraîner et 'tuner' efficacement pour améliorer le scoring, en adaptant les formats de masques fournis par Kaggle au format nécessaire pour entraîner le modèle
- Calculer, et afficher graphiquement et de manière comparative et automatique les résultats obtenus

Ceci dit, plusieurs codes déjà existants ont été récupérés et adaptés pour traiter ces différents points :

- Algorithme Matterport et Leekunhee pour le code du Mask-RCNN
- Modèle préentraîné (format .h5)
- Fonctions de décodage/encodage des masques (format RLE <-> format image)
- Fonctions de calcul du score AP
- Fonctions de suppression de l'overlap des masques prédits

Le jeu en vaut ceci dit la chandelle, puisque le but à terme est d'avoir à disposition un algorithme qui facilite le travail des médecins, chercheurs en biologie, etc., en permettant de détecter plus rapidement, et plus efficacement ces 3 différents types de cellules.
